第11期 成果発表会

『朝のスケジューリングを量子コンピュータで挑戦』

2021年10月08日

先端IT活用推進コンソーシアム
クラウド・テクノロジー活用部会
アドソル日進株式会社　荒本道隆

やりたいこと

- 朝の出勤までの行動スケジュールを最適化したい

- 「スケジュールの最適化」に、量子アニーリングを適用

- まだ、各家庭に量子コンピュータはない
- Amazon Braket を使ってみる

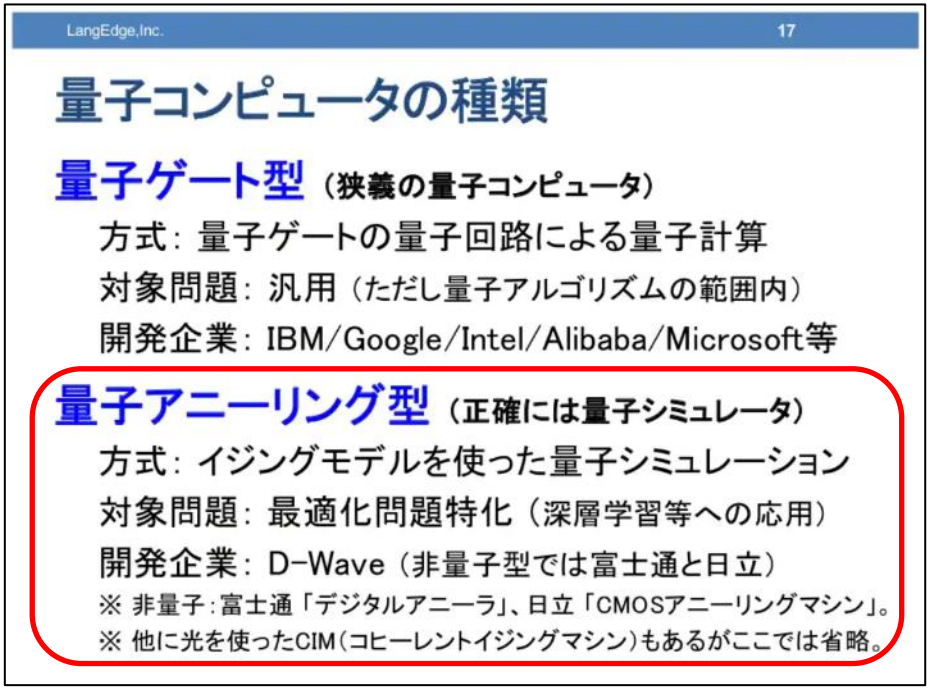
- 今回は、実用性は二の次
- **「量子コンピュータを触ってみる」が一番の目的**

参考：AWS Braket

- 個人でも簡単に量子コンピュータを使用できる
  - 3種類の量子コンピュータが選択できる

参考資料＆引用元

- 古典プログラマ向け量子プログラミング入門 [フル版]
  https://www.slideshare.net/OsSAL-org/ss-198802364
  全299ページ
  C言語に慣れ親しんでいる人に最適

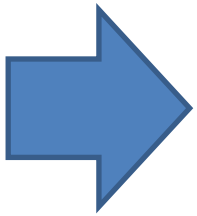


「朝の行動スケジュールの最適化」前提条件

- 今回は、朝の行動の3つだけ
  - 朝食を食べる：〇〇分
  - テレビを見る：〇〇分
  - トイレに入る：〇〇分
  - 出発
- その他の条件
  - 現在時刻
  - 交通機関の運行状況 → 出発時刻
  - 「出発時刻 － 現在時刻」に収まるよう各〇〇分を求める
  - トイレには必ず入る

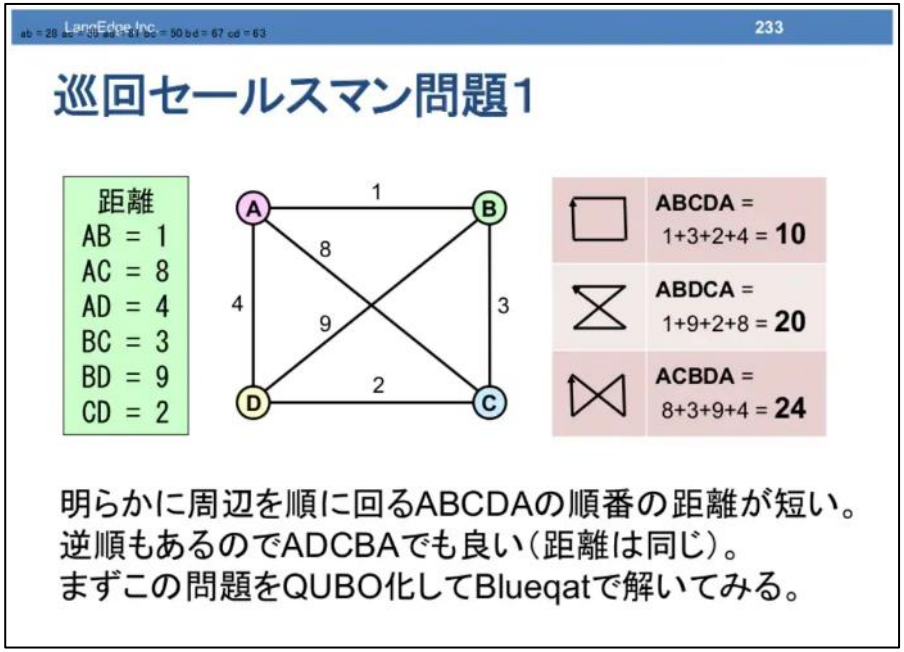
前提条件を変形

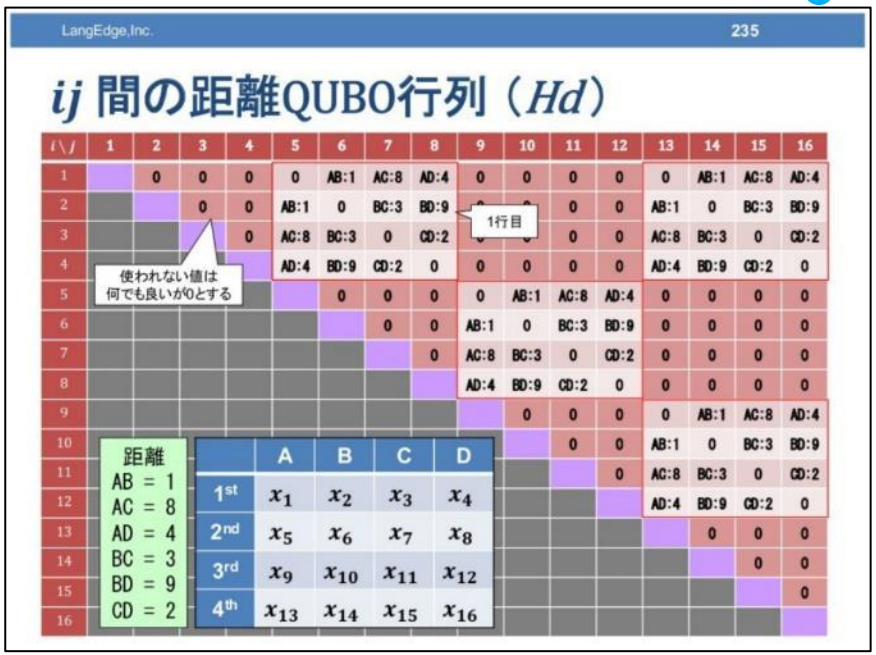
- 必要最低限の前提条件
  - 朝食を食べる：〇〇分
  - テレビを見る：〇〇分
  - トイレに入る：〇〇分
  - トイレには必ず入る
  - 残り時間は60分



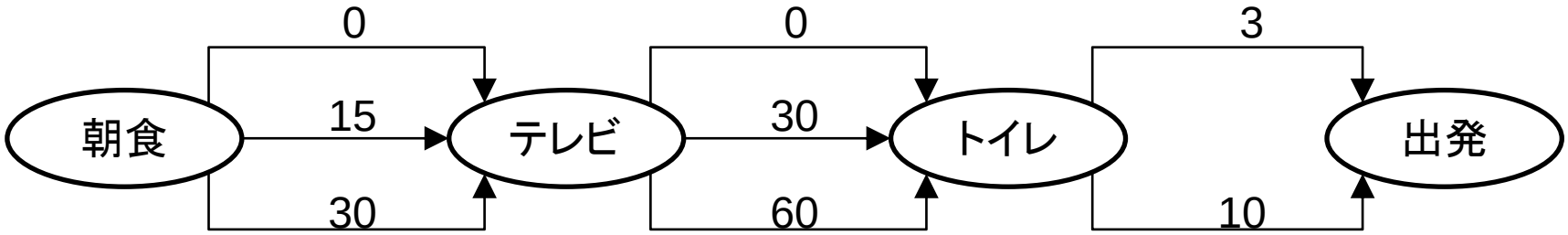
- 変形後
  - 朝食を食べる：0分（食べない）
  - 朝食を食べる：15分
  - 朝食を食べる：30分
  - テレビを見る：0分（見ない）
  - テレビを見る：30分
  - テレビを見る：60分
  - トイレに入る：3分
  - トイレに入る：10分
  - できるだけ最大値を取る
  - 60分を超えないこと

巡回セールスマン問題(p230～)を参考に





- 朝の行動スケジュール問題



めちゃめちゃ難しそう
今後、項目を増やすのが絶対大変 → 別の方法で

改めて、ナップサック問題(p266～)を参考に

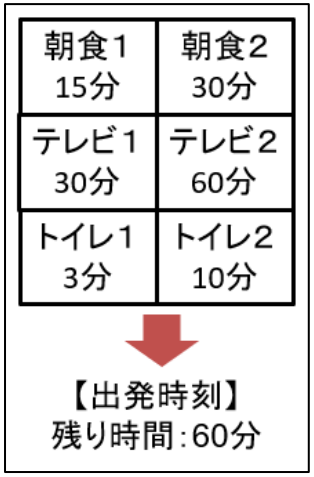
- 朝の行動スケジュール問題
  各行動の合計時間が最大、かつ、出発時刻(60分)を超えない組み合わせを求める。
  ただし、トイレの項目は必ず含むこと。
  想定解:朝食1(15分)＋テレビ1(30分)＋トイレ2(10分):合計55分

朝食1 15分	朝食2 30分
テレビ1 30分	テレビ2 60分
トイレ1 3分	トイレ2 10分

【出発時刻】
残り時間:60分

ナップサック問題(P268)を参考に定式化

- 各行動の所要時間を t_i とする
- 出発までの残り時間(60分)を L とする
- 選択されたら1になるバイナリ変数 x_i で定式化



A. コスト項 H_c:コストの合計が最大(最小)になる

$$H_c = -\Sigma t_i x_i$$

$$H_c = -(15x_1 + 30x_2 + 30x_3 + 60x_4 + 3x_5 + 10x_6)$$

B. 制限項 H_{w1}:出発までの残り時間が合計時間とイコールになる

$$H_{w1} = (L - \Sigma t_i x_i)^2$$

$$H_{w1} = (60 - (15x_1 + 30x_2 + 30x_3 + 60x_4 + 3x_5 + 10x_6))^2$$

C. 制限項 H_{w2}:トイレには必ず行く

$$H_{w2} = -(x_5 \text{ OR } x_6)$$

$$H_{w2} = -(x_5 + x_6 - x_5 x_6)$$

式全体(k_1, k_2は補正係数)

$$H = H_c + k_1 * H_{w1} + k_2 * H_{w2}$$

PyQUBOから解く

- 参考資料P270の式を変更（赤字が変更部分）
- CPUを使ってシミュレーション

```
from dimod import *
from pyqubo import *
# 変数の用意
x1, x2, x3 = Binary(“x1”), Binary(“x2”), Binary("x3")
x4, x5, x6 = Binary(“x4”), Binary(“x5”), Binary("x6")
# ハミルトニアン式
Hc = -(15*x1 + 30*x2 + 30*x3 + 60*x4 + 3*x5 + 10*x6)
Hw1 = (60 - (15*x1 + 30*x2 + 30*x3 + 60*x4 + 3*x5 + 10*x6))**2
Hw2 = -(x5 + x6 - x5*x6)
# 補正係数と全体のハミルトニアン
k1 = 10
k2 = 100
H = Hc + k1 * Hw1 + k2 * Hw2
# PyQUBOによるコンパイルとQUBO取得
model = H.compile()
qubo, offset = model.to_qubo()
#print (qubo, 'offset=', offset)
# 実行
b = BinaryQuadraticModel.from_qubo(qubo, offset)
r = SimulatedAnnealingSampler().sample(b, num_reads=100)
# 結果出力
print (r)
```

計算結果

```
   x1 x2 x3 x4 x5 x6 energy num_oc.
3   1  1  0  0  1  1 -118.0       1
6   1  1  0  0  1  1 -118.0       1
8   1  0  1  0  1  1 -118.0       1
10  1  1  0  0  1  1 -118.0       1
11  1  1  0  0  1  1 -118.0       1
15  1  0  1  0  1  1 -118.0       1
23  1  1  0  0  1  1 -118.0       1
31  1  1  0  0  1  1 -118.0       1
 :
 :
 :
95  0  1  1  0  1  0  -73.0       1
96  0  1  1  0  1  0  -73.0       1
97  0  1  1  0  1  0  -73.0       1
98  0  0  0  1  1  0  -73.0       1
99  0  0  0  1  1  0  -73.0       1
['BINARY', 100 rows, 100 samples, 6 va
riables]
```

```
# 前提条件にあてはめてみると
3  朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
6  朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
8  朝食:15分＋テレビ:30分＋トイレ:13分、合計:58分
10 朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
11 朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
15 朝食:15分＋テレビ:30分＋トイレ:13分、合計:58分
23 朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
31 朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
 :
 :
 :
95 朝食: 0分＋テレビ:60分＋トイレ: 3分、合計:63分
96 朝食:30分＋テレビ:30分＋トイレ: 3分、合計:63分
97 朝食:30分＋テレビ:30分＋トイレ: 3分、合計:63分
98 朝食:30分＋テレビ:30分＋トイレ: 3分、合計:63分
99 朝食: 0分＋テレビ:60分＋トイレ: 3分、合計:63分
```

最適解:朝食45分＋テレビ 0分＋トイレ13分、合計:58分
最適解:朝食15分＋テレビ30分＋トイレ13分、合計:58分

想定解以上のものが2つも出てきた

量子コンピュータで解く

```
# 量子コンピュータを使用するように変更（赤字が変更部分）
from braket.ocean_plugin import BraketSampler, BraketDWaveSampler
from dwave.system.composites import EmbeddingComposite
s3_folder = (f“amazon-braket-xxxxxxxxxxxx”, “Your-Folder-Name”)　#自分のS3のBucket名を指定
from dimod import *
from pyqubo import *
# 変数の用意
x1, x2, x3 = Binary("x1"), Binary("x2"), Binary("x3")
x4, x5, x6 = Binary("x4"), Binary("x5"), Binary("x6")
# ハミルトニアン式
Hc = -(15*x1 + 30*x2 + 30*x3 + 60*x4 + 3*x5 + 10*x6)
Hw1 = (60 - (15*x1 + 30*x2 + 30*x3 + 60*x4 + 3*x5 + 10*x6))**2
Hw2 = -(x5 + x6 - x5*x6)
# 補正係数と全体のハミルトニアン
k1 = 10
k2 = 100
H = Hc + k1 * Hw1 + k2 * Hw2
# PyQUBOによるコンパイルとQUBO取得
model = H.compile()
qubo, offset = model.to_qubo()
#print (qubo, 'offset=', offset)
# 実行
b = BinaryQuadraticModel.from_qubo(qubo, offset)
#r = SimulatedAnnealingSampler().sample(b, num_reads=100)
# run BQM: solve with the D-Wave 2000Q device
sampler = BraketDWaveSampler(s3_folder,'arn:aws:braket:::device/qpu/d-wave/DW_2000Q_6')
sampler = EmbeddingComposite(sampler)
r = sampler.sample(b, num_reads=100)
# 結果出力
print (r)
```

量子コンピュータでの計算結果

	x1	x2	x3	x4	x5	x6	energy	num_oc.	chain_.
0	1	1	0	0	1	1	-118.0	3	0.0
1	1	0	1	0	1	1	-118.0	3	0.0
2	0	1	1	0	1	0	-73.0	17	0.0
3	0	0	0	1	1	0	-73.0	3	0.0
4	0	1	1	0	0	0	-60.0	22	0.0
5	0	0	0	1	0	0	-60.0	4	0.0
6	1	0	1	0	0	1	95.0	5	0.0
7	1	1	0	0	0	1	95.0	3	0.0
8	0	0	0	1	0	1	830.0	5	0.0
9	0	1	1	0	0	1	830.0	10	0.0
10	1	0	1	0	1	0	1292.0	4	0.0
11	0	0	0	1	1	1	1517.0	2	0.0
12	0	1	1	0	1	1	1517.0	1	0.0
13	1	1	1	0	0	0	2175.0	5	0.0
14	1	0	0	1	0	0	2175.0	2	0.0
15	1	1	0	0	0	0	2205.0	1	0.0
16	1	0	1	0	0	0	2205.0	6	0.0
17	0	0	1	0	1	1	2747.0	1	0.0
18	1	1	1	0	1	0	3062.0	2	0.0
19	0	1	0	1	1	0	10697.0	1	0.0

['BINARY', 20 rows, 100 samples, 6 variables]

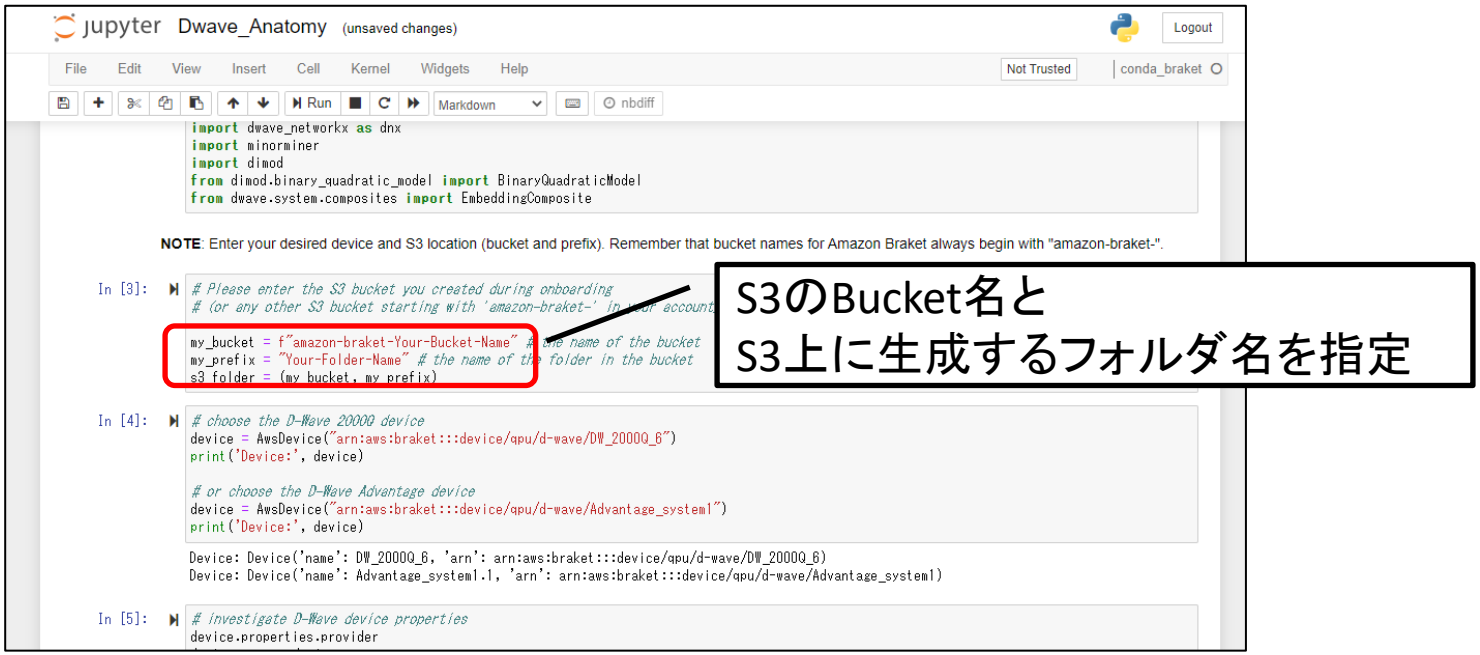
前提条件にあてはめてみると

0 朝食:45分＋テレビ: 0分＋トイレ:13分、合計:58分
1 朝食:15分＋テレビ:30分＋トイレ:13分、合計:58分
2 朝食:30分＋テレビ:30分＋トイレ: 3分、合計:63分
3 朝食: 0分＋テレビ:60分＋トイレ: 3分、合計:63分
4 朝食:30分＋テレビ:30分＋トイレ: 0分、合計:60分
5 朝食: 0分＋テレビ:60分＋トイレ: 0分、合計:60分
6 朝食:15分＋テレビ:30分＋トイレ:10分、合計:55分
7 朝食:45分＋テレビ: 0分＋トイレ:10分、合計:55分
8 朝食: 0分＋テレビ:60分＋トイレ:10分、合計:70分
9 朝食:30分＋テレビ:30分＋トイレ:10分、合計:70分
10 朝食:15分＋テレビ:30分＋トイレ: 3分、合計:48分
11 朝食: 0分＋テレビ:60分＋トイレ:13分、合計:73分
12 朝食:30分＋テレビ:30分＋トイレ:13分、合計:73分
13 朝食:45分＋テレビ:30分＋トイレ: 0分、合計:75分
14 朝食:15分＋テレビ:60分＋トイレ: 0分、合計:75分
15 朝食:45分＋テレビ: 0分＋トイレ: 0分、合計:43分
16 朝食:30分＋テレビ:30分＋トイレ: 0分、合計:60分
17 朝食: 0分＋テレビ:30分＋トイレ:13分、合計:43分
18 朝食:45分＋テレビ:30分＋トイレ: 3分、合計:78分
19 朝食:30分＋テレビ:60分＋トイレ: 3分、合計:93分

最適解:朝食45分＋テレビ 0分＋トイレ13分、合計:58分
最適解:朝食15分＋テレビ30分＋トイレ13分、合計:58分

参考：AWS Braketの動かし方

- 「Amazon Braket」を開く
- 「Notebooks」→「Create notebook instance」
  - 作成されるまで数分かかる
- サンプルを選択
- 自分のS3のBucket名を指定する
- メニューの「Cell」→「Run All」で実行する
  - サンプルは結構複雑な処理 → **結構課金される**



参考：AWS Braketの利用料金

- D-wave 2000Qの場合
  - $0.3 + $0.00019 * shot数
- 量子アニーリングでは、何度も試行して分布を調べる
  - 例：100回試行する場合
    - $0.3 + $0.00019 * 100 = $0.319 = 約35円
  - 毎分再評価すると、1時間あたり
    - 約35円 x 60回 ＝ 約2,100円／時

Pricing Tables

Quantum Computers

Hardware Provider	QPU family	Per-task price	Per-shot price
D-Wave	2000Q	$0.30000	$0.00019
D-Wave	Advantage	$0.30000	$0.00019
IonQ	IonQ device	$0.30000	$0.01000
Rigetti	Aspen-9	$0.30000	$0.00035

『Amazon Braket Pricing』

今回のコードの実行にかかった金額

主にサンプルを3つ実行したせい

- 前日の「今月の請求書」

Braket		$17.21
US West (Oregon)		**$17.21**
Amazon Braket CompleteTask		$2.81
$0.00019 per-shot for D-Wave-2000Q in US West (Oregon) Region	11,780.000 Quantum-Shot	$2.24
$0.00019 per-shot for D-Wave-Advantage_system in US West (Oregon) Region	3,010.000 Quantum-Shot	$0.57
Amazon Braket Task		$14.40
$0.30 per-task for D-Wave-2000Q in US West (Oregon) Region	44.000 Quantum-Task	$13.20
$0.30 per-task for D-Wave-Advantage_system in US West (Oregon) Region	4.000 Quantum-Task	$1.20

- 1回（100試行）だけ実行した、翌日の「今月の請求書」

Braket		$17.53	+$0.32
US West (Oregon)		**$17.53**	
Amazon Braket CompleteTask	+100 Q-Shot	$2.83	
$0.00019 per-shot for D-Wave-2000Q in US West (Oregon) Region	11,880.000 Quantum-Shot	$2.26	
$0.00019 per-shot for D-Wave-Advantage_system in US West (Oregon) Region	3,010.000 Quantum-Shot	$0.57	
Amazon Braket Task	+1 Q-Task	$14.70	
$0.30 per-task for D-Wave-2000Q in US West (Oregon) Region	45.000 Quantum-Task	$13.50	
$0.30 per-task for D-Wave-Advantage_system in US West (Oregon) Region	4.000 Quantum-Task	$1.20	

- 上記に加えて、Notebookの利用料金（0.14$/GB-Mo）もかかる

やってみた感想

- AWS Braketのサンプルを適当に3つ動かしてみたら
  - 実行にちょっと時間がかかるなぁ、と数分間眺めていたら、
  - 翌日、請求額を確認すると$11.93（約1,312円）になっていた
- ナップサック問題をベースにすることで
  - 想定していた以上の最適解が出た
  - 項目の追加が容易
  - 複雑な条件が出てきた時、数式にできるのだろうか？
  - 条件をクリアしていない結果も返ってくるので、チェックする実装も必要
- 実用になるのか？
  - 35円／回、毎分再評価すると2,100円／時、はかかりすぎ
    - きっかけがあった時だけ再計算したい
    - 数百円／時、なら許容範囲？
    - 東芝の「シミュレーテッド分岐マシン」（AWS上）なら$3.06/時間
      https://aws.amazon.com/marketplace/pp/prodview-f3hbaz4q3y32y
  - 朝食は、「毎日食べる」か「毎日食べない」が体重維持には良いらしい
  - テレビを見るかどうかは、やっているニュース次第
  - トイレにかかる時間は、自分ではコントロールできない

まとめ

- 理由をつけて、量子コンピュータを触ってみる：達成
  - なんとなくデモを動かしてみるよりも理解が深まった
  - 超高額なハードウエアを利用するので、それなりに費用がかかる
    - 実機をレンタルするよりは、はるかに安価だけど
- 実際に運用できそうか？：しばらくは無理そう
  - 量子コンピュータの費用が結構かかる
  - 優先するものが、人ごとに、日によっても異なりそう
  - 外部／内部の状況をもっと取り込んで反映させたい
    - テレビで放送中のニュースは、興味があるか？
    - 便意の強度
- 今回の実証実験は意味があったか？：もちろん、意味があった
  - やってみよう → 具体化しなきゃならない
    - 定式化してみるだけでも、色んなものが見えてきた
    - 定式化したものは、量子コンピュータじゃなくても実行できる
  - 補正係数は、どうやって決める？
    - 補正係数 ＝ 優先度 → 人ごとに違いそう

ありがとうございました

http://aitc.jp

https://www.facebook.com/aitc.jp

ハルミン
AITC非公式イメージキャラクター