Na początku umieszczam pytania, dalej są informacje uszczegóławiające, do których pytania się odnoszą.

1. Pytania

1. Czy jest jakiś przyjęty model, wzorzec, metoda pracy z problemami typu QUBO/Ising'a i wprowadzania ich na komputer D'Wave tak, aby rozwiązania
2. Biblioteka tworząca CSP ma ograniczenia, w szczególności gdy dostaje constrainty z dużą ilością zmiennych wymaga bardzo małej różnicy energii względem "ground", gdy zwiększyłem maksymalny czas z 4 do 8 biblioteka zwyczajnie odmówiła stworzenia obiektu tłumacząc, że żadna z jej fabryk nie jest tego w stanie zrobić. Czy efektywne jest wykorzystanie takich bibliotek, czy może lepiej jest jednak ręcznie tworzyć problemy QUBO i eksperymentować z wartościami mnożników dla poszczególnych constraintów przy ich sumowaniu.
3. Czy może być tak, że dobrze zaprojektowany problem QUBO zwraca złe wyniki na komputerze kwantowym ze względu na architekturę komputera kwantowego (Chimera)? Czy może być inna przyczyna?
4. Przy transformacji problemów do postaci QUBO wszystkie zmienne przyjmujące wartości z zadanego zbioru (np. w poniższym przykładzie czas ze zbioru {1,2,3,4} muszą być reprezentowane jako one-hot (4 kubity - jedynka na jednym z nich oznacza że zmienna ma taką wartość, reszta to zera). Czy istnieje możliwość obejścia tego?
5. Czy podejście do rozwiązywania problemów na D-Wave, które poniżej przedstawiam w ogóle ma sens, czy idę w dobrą stronę, tylko potrzebuję zmienić kilka mniejszych bądź większych rzeczy, czy powinienem zupełnie zmienić tor działania?
6. Ile to jest "wystarczająco dużo" uruchomień jednego problemu w postaci QUBO na D'Wave? "Wystarczająco dużo" - tak dużo, żeby wyszły "dobre" bez względu na ewentualną fizyczną ułomność maszyny.

2. Reprezentacja problemu job shop

Póki co zajmuję się tylko spełnieniem constraint'ów, nie optymalizuję jeszcze czasu.
Problem job-shop reprezentuję następująco (na podstawie https://arxiv.org/pdf/1506.08479.pdf):
Rozmiar problemu to $OP * M * T$, gdzie

- $OP - calkowita\ liczba\ operacji\ we\ wszystkich\ jobach$
- $M - całkowita\ liczba\ maszyn$
- $T - maksymalna\ ilość\ czasu$

Jeśli kubit o numerze $op + OP * m + OP * M * t$ przyjmuje wartość "1", to operacja numer op zaczyna się na maszynie o numerze m w czasie t (wszystko indeksowane od zera).
Najlepiej widać tą reprezentację na przykładzie:

$jobs = [[2, 1], [1, 2, 1]]$ $(2\ joby,\ pierwszy\ ma\ dwie\ operacje\ o\ długościach\ 2\ oraz\ 1,$
$drugi\ ma\ 3\ operacje\ o\ długościach\ 1, 2, 1)$

$OP = 5$

$M = 2$

$T = 4$

Nr operacji	**0**	**1**	**2**	**3**	**4**	**0**	**1**	**2**	**3**	**4**
Nr maszyny	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**
0	0	1	2	3	4	5	6	**7**	8	9
1	10	11	12	13	14	15	16	17	18	19
2	20	21	22	23	24	25	26	27	28	29
3	30	31	32	33	34	35	36	37	38	39
Czas										

Kubit numer 27 ma wartość “1” gdy operacja numer 2 zaczyna się na maszynie nr 1 w momencie czasowym “2”

3. Przekształcenie do QUBO

Na początku problem jest przekształcany do postaci CSP (Constraint Satisfaction Problem). Constrainty dodawane są z użyciem obiektów klasy dwavebinarycsp.ConstraintSatisfactionProblem na podstawie danych wejściowych, w powyższym przykładzie dokładnie jeden z kubitów 0,5,10,15,20,25,30,35 ma mieć wartość “1”, gdyż każda operacja musi zacząć się dokładnie raz.
Constrainty są następujące:

- Każda operacja musi zacząć się dokładnie raz
- Na jednej maszynie może w jednym momencie wykonywać tylko jedna operacja
- Operacja następująca w danym jobie musi zacząć się nie wcześniej niż skończy się operacja poprzedzająca

Po stworzeniu obiektu CSP jest on przekształcany do QUBO z użyciem najpierw bqm (binary quadratic model), potem przekształcany do QUBO (automatycznie: bqm.to_qubo())

```
bqm = dwavebinarycsp.stitch(job_shop_problem.csp, max_graph_size=16, min_classical_gap=0.6)
```

4. Uruchomienie problemu QUBO na komputerze kwantowym

Uruchomienie bez rozdzielenie

Problem tak przygotowany uruchamiam na komputerze D-Wave następującym wywołaniem:

```
response = EmbeddingComposite(DWaveSampler()).sample_qubo(Q, num_reads=200)
```

Otrzymuję w wyniku kilkadziesiąt rozwiązań, z których żadne nie spełnia wcześniej założonych constraintów (sprawdzane funkcją csp.check()

Rozdzielenie problemu za pomocą QBSolv

Stosuję bibliotekę QBSolv(), która rozdziela problem QUBO na mniejsze, wywołuje pojedynczo na komputerze D-Wave i składa wyniki z powrotem

```
sampler = EmbeddingComposite(DWaveSampler())
response = QBSolv().sample_qubo(Q, solver=sampler, solver_limit=30)
```

Wyniki są lepsze. Dla problemu z czterema operacjami, dwoma maszynami i limitem czasu 4 wszystkie wyniki spełniają constrainty, aczkolwiek po dodaniu jeszcze jednej operacji już nie wszystkie wyniki są dobre.