Installazione (Win10)

Seguono i passi per creare un environment python in cui usare sia la libreria `pyqubo`, sia l'attuale SDK D-Wave ufficiale:

```
conda install conda=24.1.2
conda create --name dwave_pyqubo python=3.11
conda activate dwave_pyqubo
pip install pyqubo
pip install dwave-ocean-sdk
dwave auth login
dwave auth get
dwave config create --auto-token
```

Generano il folder:

```
C:\Users<user-name>\AppData\Local\dwavesystem\dwave-pyqubo
```

Uso tramite Visual Studio Code

Selezionare l'interprete corretto, disponibile nell'environment `dwave_pyqubo':

```
Ctrl-Shft-P > Python: Select interpreter > Python 3.11.8 ('dwave_pyqubo')
```

Inspector

Strumento per apprezzare visualmente il lavoro svolto dall'algoritmo di Minor embedding. Esso mappa una *clique* che rappresenta un modello QUBO nel grafo Pegasus i cui nodi sono i qbit della Qpu.

La librerie di proprietaria D-Wave si installa seguendo dwave-inspector:

```
pip install dwave-inspector
pip install dwave-inspectorapp --extra-index-url=https://pypi.dwavesys.com/simple
```

File di configurazione dwave.conf

Non ricordo più in che punto del processo di installazione, ma è stato necessario "indovinare" la struttura del file `dwave.conf` .

Il file è necessario per usare i solver ibridi, quindi la Qcpu.

Il tempo concesso dipende o dal proprio piano di abbonamento al *cloud* D-Wave, o dall'aver reso pubblico il repository Github con i sorgenti che usano l'SDK D-Wave, librerie proprietarie incluse.

La struttura del file `dwave.conf` che ho ricavato dando un'occhiata (anche) a D-Wave Ocean Software Documentation è:

> [defaults]
> token = DEV-.....
>
> [default-solver]
> client = hybrid
> #solver = {"num_qubits__gt": 5000}
>
> #[hybrid]
> #client =
>
> [europe]
> region = eu-central-1

Il `token = DEV-...` è assegnato da D-Wave all'account da aprire su https://cloud.dwavesys.com.