

```
In [29]: import plotly.io as pio
pio.renderers.default = "notebook"

In [30]: from dwave.system import DWaveSampler, EmbeddingComposite
from schedule import *
import neal
import plotly.express as px
from tabu import TabuSampler
from dwave_qbsolv import QBSolv
from anneal_solver import solveilog_dwave
```

Job-Shop-Scheduling mit Annealer

```
In [31]: #Flag damit nicht unbeabsichtigt der quantumcomputer verwendet wird
use_quantum=True
```

Erstelle den Ablaufplan

```
In [32]: #maximale Zeit
s = Schedule(time_max=10)
# maximal 3 maschinen
# bitte nur benötigte anzahl der maschinen angeben, wir haben keinen preprocessing step um unnötige maschinen zu entfernen
s.build_machines(7)

#Jobs
# Job 1 (1,2,3) bedeutet auf der maschine 1 wird die arbeit 2 seiteinheiten am stück durchgeführt und 3 mal wiederholt
# es sollte wenn möglich nur die dauer auf 1 begrenzt sein, da wir keine zeit hatten die anderen optionen gut zu testen
# frust möglichsterpunkte könnten zwar gesetzt werden, dies bricht zurzeit noch einige feature und sichtbarkeit
s.create_job((0: (1, 1), 2: (1, 1)), parallel_operations=1)
s.create_job((3: (1, 2), 0: (1, 1)), parallel_operations=1)
s.create_job((4: (1, 1), 1: (1, 1), 2: (1, 1)), parallel_operations=1)
s.create_job((6: (1, 1), 2: (1, 1)), parallel_operations=1)

print(s)

-----
Arbeitsplan
Job 0 Operationen:
Maschine(nr=0, name='m0', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
Maschine(nr=2, name='m2', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1

Job 1 Operationen:
Maschine(nr=3, name='m3', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 2
Maschine(nr=0, name='m0', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1

Job 2 Operationen:
Maschine(nr=4, name='m4', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
Maschine(nr=0, name='m0', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1

Job 3 Operationen:
Maschine(nr=5, name='m5', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
Maschine(nr=1, name='m1', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
Maschine(nr=2, name='m2', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1

Job 4 Operationen:
Maschine(nr=6, name='m6', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
Maschine(nr=2, name='m2', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1
```

Qubostraf Terme und Berechnung des QUBO

```
In [33]: qubo=Qubo(s)

#start term
qubo.penalty_terms[1]=1
# nur eine maschine/job pro job/maschine gleichzeitig
qubo.penalty_terms[2]=1
qubo.penalty_terms[3]=1

# 4 hat keine bedeutung, war ursprünglich für die deadlines vorgesehen
# da wir diese aber ohne strafterme lösen, ist dieses feld leer

# Just in time
qubo.penalty_terms[5]=0

qubo.calculate_qubo()
print(f"qubits: {len(qubo.h)}")

# qubo berechnung falls mehreren maschinen oder jobs gleichzeitig verwendet werden sollten - (dies muss im schedule)
qubo.calculate_qubo_virtual()

qubits: 374
```

Visualisierung des QUBOs

```
In [34]: qubo.plot_qubo()
```



Simulation mit QBsolve

```
In [37]: response = QBSolve().sample_qubo(qubo.J, num_repeats=1000)
sol1 = response.samples[0]
*temp, _ = qubo.interpret_solution_dict(sol1)
print(temp)
qubo.plot_solution()

[[10, 22, 76, 107, 116, 199, 243, 248, 284, 321, 365], [state(j:0,m:0,t:1), state(j:0,m:2,t:3), state(j:1,m:0,t:1), state(j:1,m:2,t:3), state(j:2,m:0,t:1), state(j:2,m:2,t:3), state(j:3,m:0,t:1), state(j:3,m:2,t:3), state(j:4,m:0,t:1), state(j:4,m:2,t:3), state(j:5,m:0,t:1), state(j:5,m:2,t:3), state(j:6,m:0,t:1), state(j:6,m:2,t:3)], [state(j:0,m:0,t:10), state(j:0,m:2,t:9), state(j:1,m:0,t:10), state(j:1,m:2,t:9), state(j:2,m:0,t:10), state(j:2,m:2,t:9), state(j:3,m:0,t:10), state(j:3,m:2,t:9), state(j:4,m:0,t:10), state(j:4,m:2,t:9), state(j:5,m:0,t:10), state(j:5,m:2,t:9), state(j:6,m:0,t:10), state(j:6,m:2,t:9)], [-14.0, 1: -14, 2: 2, 3: 0, 4: 0, 5: 0.0, 14]]
```



Erhöhen des Strafterms welche bestrafst falls ein Operation zu häufig oder nicht oft genug gestartet wird

```
In [41]: qubo = Qubo(s)

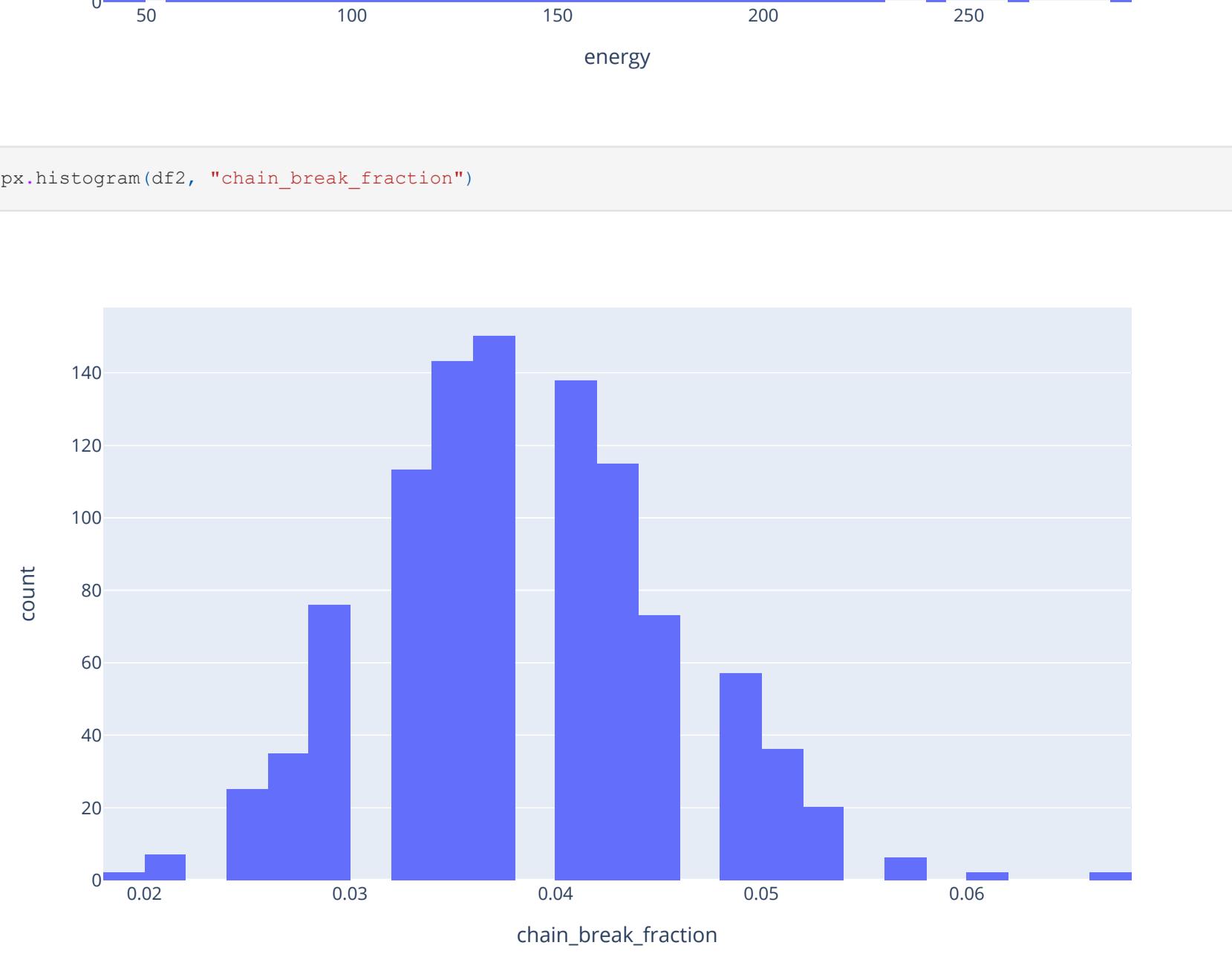
#start term
qubo.penalty_terms[1] = 5
# nur eine maschine/job pro job/maschine gleichzeitig
qubo.penalty_terms[2] = 1
qubo.penalty_terms[3] = 1

# 4 hat keine bedeutung, war ursprünglich für die deadlines vorgesehen
# da wir diese aber ohne strafterme lösen, ist dieses feld leer

# Just in time
qubo.penalty_terms[5] = 0

qubo.calculate_qubo()
```

```
In [42]: if use_quantum:
    answer2 = solveilog_dwave(qubo, samples=10000)
    *temp2, _ = qubo.interpret_solution_dict(answer2)
    print(temp2)
    qubo.plot_solution()
```



```
In [43]: df1=answer.to_pandas_dataframe()
df1
```

```
Out[57]: state(j:0,m:0,t:0) state(j:0,m:0,t:1) state(j:0,m:0,t:2) state(j:0,m:0,t:3) state(j:0,m:0,t:4) state(j:0,m:0,t:5)
```

job_id	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...
995	0	0	0	0	0	0
996	0	0	0	0	0	0
997	0	1	0	0	0	0
998	0	0	0	0	0	0
999	0	0	1	0	0	0

```
1000 rows x 377 columns
```

```
In [56]: px.histogram(df1, "energy")
```



```
In [59]: px.histogram(df1, "chain_break_fraction")
```



```
In [53]: df2=answer2.to_pandas_dataframe()
df2
```

```
Out[54]: px.histogram(df2, "energy")
```



```
In [58]: px.histogram(df2, "chain_break_fraction")
```



```
In [60]: # uncoment to export it as html
#jupyter nbconvert --to html example4.ipynb
```

```
[NbConvertApp] WARNING | kernel_spec_manager_class' not recognized by 'NbConvertApp'.
[D:\NbConvertApp] CONVERTING | notebook example4.ipynb to_html
D:\NbConvertApp| ERROR | kernel_spec_manager_class' not recognized by 'NbConvertApp'.
[D:\NbConvertApp] ERROR | kernel_spec_manager_class' not able to be represented.
  warn("Your element was not able to be represented."
[NbConvertApp] Writing 5722091 bytes to example4.html
```

```
In [ ]:
```

```
In [ ]:
```