

```
In [1]:  
import plotly.io as pio  
pio.renderers.default = "notebook"  
  
In [2]:  
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 solvelog_dwave
```



Job-Shop-Scheduling mit Annealer

```
In [3]:  
#Flag damit nicht unbeabsichtigt der quantumcomputer verwendet wird  
use_quantum=True
```

Erstelle den Ablaufplan

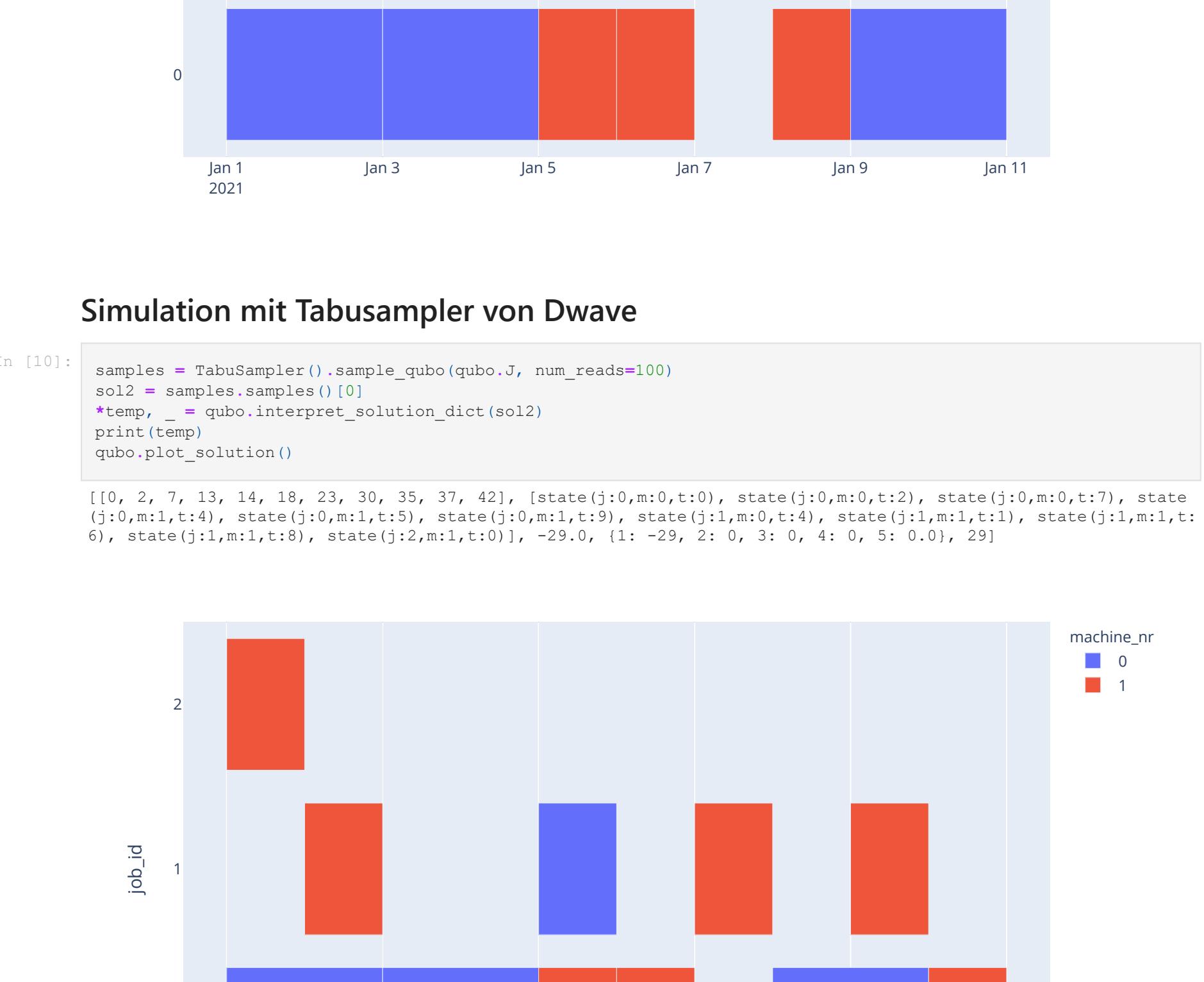
```
In [4]:  
#maximale Zeit  
s = Schedule(time_max=10)  
# maximal 3 maschinen  
# bitte nur benötigte anzahl der maschinen angeben, wir haben keinen preprocessing step um unnötige maschinen l  
s.build_machines(2)  
  
#Jobs  
# 1:(2,3) bedeutet auf der maschine 1 wird die arbeit 2 zeiteinheiten am 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  
# frustig möglicherzeitpunkte können zwar gesetzt werden, dies bringt zurzeit noch einige feature und so  
s.create_job({1:(1,3),0:(2,3)},parallel_operations=1)  
s.create_job({1:(1,3),0:(1,1)}  
s.create_job({1:(1,1)}, deadline=2)  
  
print(s)  
  
-----  
Arbeitsplan  
Job 0 Operationen:  
Maschine(nr=1, name='m1', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 3  
Maschine(nr=0, name='m0', parallel_operations=1, start_time=0) Dauer: 2 Anzahl: 3  
  
Job 1 Operationen:  
Maschine(nr=1, name='m1', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 3  
Maschine(nr=0, name='m0', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1  
  
Job 2 Operationen:  
Maschine(nr=1, name='m1', parallel_operations=1, start_time=0) Dauer: 1 Anzahl: 1  
deadline: 2  
-----
```

Qubostraf Terme und Berechnung des QUBO

```
In [5]:  
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 mehrere Maschinen oder Jobs gleichzeitig verwendet werden sollten - (dies muss im sche  
# qubo.calculate_gubo_virtuell()  
  
qubits: 44
```

Visualisierung des QUBOs

```
In [6]:  
qubo.plot_qubo()
```



```
In [7]:  
qubo.plot_connections_qubo_states(270)
```

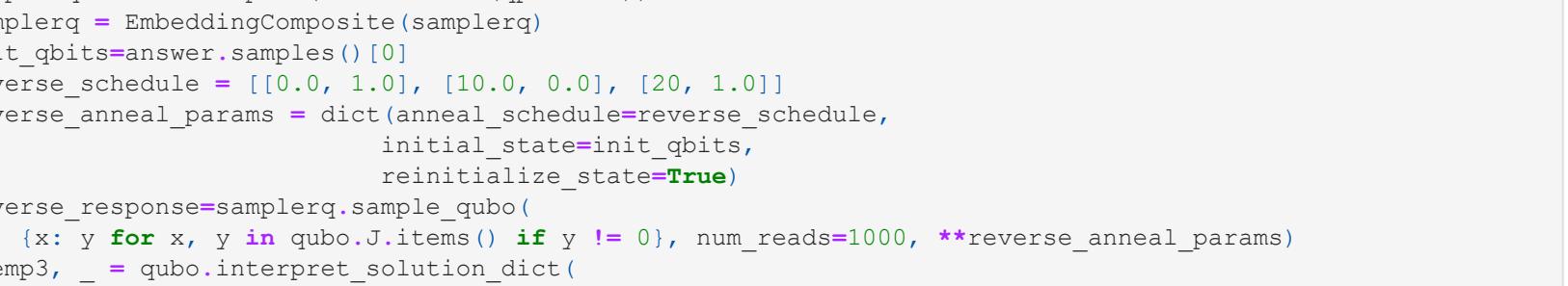
```
Out[7]:
```



Simulation mit QBSolve

```
In [9]:  
response = QBSolv().sample_qubo(qubo.J, num_repeats=100)  
sol3 = response.samples()[0]  
*temp, _ = qubo.interpret_solution_dict(sol3)  
print(temp)  
qubo.plot_solution()
```

```
[{0, 2, 8, 13, 14, 16, 20, 31, 32, 38, 43}, {state(j:0,m:0,t:0), state(j:0,m:0,t:2), state(j:0,m:0,t:8), state(j:0,m:1,t:4), state(j:0,m:1,t:5), state(j:0,m:1,t:7), state(j:0,m:1,t:9), state(j:0,m:1,t:10), state(j:1,m:0,t:5), state(j:1,m:0,t:7), state(j:1,m:0,t:8), state(j:1,m:1,t:2), state(j:1,m:1,t:3), state(j:1,m:1,t:9), state(j:2,m:1,t:0)}], -29.0, {1: -29, 2: 0, 3: 0, 4: 0, 5: 0.0}, 29]
```



Simulation mit Tabusampler von Dwave

```
In [10]:  
samples = TabuSampler().sample_qubo(qubo.J, num_reads=100)  
sol2 = samples.samples()[0]  
*temp, _ = qubo.interpret_solution_dict(sol2)  
print(temp)  
qubo.plot_solution()
```

```
[{0, 2, 7, 13, 14, 18, 23, 30, 35, 37, 42}, {state(j:0,m:0,t:0), state(j:0,m:0,t:2), state(j:0,m:0,t:7), state(j:0,m:1,t:4), state(j:0,m:1,t:5), state(j:0,m:1,t:7), state(j:0,m:1,t:9), state(j:1,m:0,t:8), state(j:1,m:1,t:1), state(j:1,m:1,t:6), state(j:1,m:1,t:8), state(j:2,m:1,t:0), state(j:1,m:1,t:3), state(j:1,m:1,t:5), state(j:1,m:1,t:4)}, -29.0, {1: -29, 2: 0, 3: 0, 4: 0, 5: 0.0}, 29]
```



Simulation mit Simulated Annealer

```
In [11]:  
sampler = neal.SimulatedAnnealingSampler()  
sampleset = sampler.sample_qubo(qubo.J, num_reads=100)  
*temp, _ = qubo.interpret_solution_dict(sampleset[0])  
print(temp)  
qubo.plot_solution()
```

```
[{2, 4, 7, 10, 15, 18, 20, 31, 33, 34, 42}, {state(j:0,m:0,t:0), state(j:0,m:0,t:3), state(j:0,m:0,t:5), state(j:0,m:1,t:2), state(j:0,m:1,t:6), state(j:0,m:1,t:7), state(j:0,m:1,t:8), state(j:1,m:0,t:8), state(j:1,m:1,t:1), state(j:1,m:1,t:4), state(j:1,m:1,t:5), state(j:2,m:1,t:0)}, -29.0, {1: -29, 2: 0, 3: 0, 4: 0, 5: 0.0}, 29]
```


Quantum Computer von Dwave

```
In [12]:  
if use_quantum:  
    samplerq = DWaveSampler(solver=dict(epu=True))  
    samplerq = EmbeddingComposite(samplerq)  
    init_gbts=samples[0][0]  
    *temp, _ = qubo.interpret_solution_dict(init_gbts)  
    print(temp)  
    qubo.plot_solution()
```

```
[{3, 5, 11, 16, 17, 27, 42, 30, 33, 34}, {state(j:0,m:0,t:0), state(j:0,m:0,t:3), state(j:0,m:0,t:5), state(j:0,m:1,t:2), state(j:0,m:1,t:7), state(j:0,m:1,t:8), state(j:1,m:0,t:8), state(j:1,m:1,t:1), state(j:1,m:1,t:4), state(j:1,m:1,t:5), state(j:2,m:1,t:0)}, -29.0, {1: -29, 2: 0, 3: 0, 4: 0, 5: 0.0}, 29]
```


Reverse Annealing mit Dwave

```
In [13]:  
if use_quantum:  
    samplerq = DWaveSampler(solver=dict(epu=True))  
    samplerq = EmbeddingComposite(samplerq)  
    init_gbts=samples[0][0]  
    *temp, _ = qubo.interpret_solution_dict(init_gbts)  
    reverse_schedule = [[0,0, 1,0], [10,0, 0,0], [20, 1,0]]  
    reverse_anneal_params = dict(anneal_schedule=reverse_schedule,  
                                initial_state=init_gbts,  
                                reinitialize_state=True)  
    reverse_response=samplerq.sample_qubo(  
        (x: y for x, y in qubo.J.items() if y != 0), num_reads=1000, **reverse_anneal_params)  
    *temp, _ = qubo.interpret_solution_dict(  
        (x: y for x, y in reverse_response[0].items()))  
    print(temp)  
    qubo.plot_solution()
```

```
[{0, 3, 5, 11, 16, 17, 27, 42, 30, 33, 34}, {state(j:0,m:0,t:0), state(j:0,m:0,t:3), state(j:0,m:0,t:5), state(j:0,m:1,t:2), state(j:0,m:1,t:7), state(j:0,m:1,t:8), state(j:1,m:0,t:8), state(j:1,m:1,t:1), state(j:1,m:1,t:4), state(j:1,m:1,t:5), state(j:2,m:1,t:0)}, -29.0, {1: -29, 2: 0, 3: 0, 4: 0, 5: 0.0}, 29]
```

