frontiers

A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer

Sebastian Feld [1,*], **Christoph Roch** [1], **Thomas Gabor** [1], **Christian Seidel** [2], **Florian Neukart** [3], **Isabella Galter** [2], **Wolfgang Mauerer** [4], **Claudia Linnhoff-Popien** [1]

[1] *LMU Munich, Mobile and Distributed Systems Group, Munich, Germany*
[2] *Volkswagen Data:Lab, Munich, Germany*
[3] *Volkswagen Group of America, San Francisco, USA*
[4] *OTH Regensburg, Regensburg, Germany*

Correspondence*:
Sebastian Feld, Oettingenstr. 67, 80538 Munich, Germany
sebastian.feld@ifi.lmu.de

1 APPENDIX

```
number of clusters: 5

00100000000001000000010000000000001000000000000100000001000000000000100000010000000100001000000000
-4922.00000 Energy of solution
4 Number of Partitioned calls, 1 output sample
0.01600 seconds of classic cpu time

00000000100000000010000000000001001000000000000000000010010000000100000000000000100000000000100000
-7646.00000 Energy of solution
4 Number of Partitioned calls, 1 output sample
0.04600 seconds of classic cpu time

00010000000000001000000000000010000000000100000000000001000000100000000000000000100000000000010000
-12552.00000 Energy of solution
2 Number of Partitioned calls, 1 output sample
0.01600 seconds of classic cpu time

00000010000000000100000001000000000000100010000000010000000000010000001000000000000001000000000000001
-6903.00000 Energy of solution
8 Number of Partitioned calls, 1 output sample
0.03100 seconds of classic cpu time

00010000000001000000000001000000000000010000000000001000000000000000000100000000001000000000010000000
-16253.00000 Energy of solution
32 Number of Partitioned calls, 1 output sample
0.12500 seconds of classic cpu time

Total distance 557
```

Listing 1. Output of locally used QBSolv.

```
738180 function calls (661690 primitive calls) in 1.474 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall      filename:lineno(function)
5       0.262    0.052    0.262    0.052        {dwave_qbsolv.qbsolv_binding.run_qbsolv}
51      0.066    0.001    1.258    0.025        __init__.py:1(<module>)
364     0.058    0.000    0.058    0.000        {imp.find_module}
37752   0.056    0.000    0.056    0.000        binary_quadratic_model.py:484(add_interaction)
5       0.043    0.009    0.044    0.009        TSPPreparer.py:56(generate_adjacency_matrix)
78703   0.040    0.000    0.060    0.000        records.py:438(__getattribute__)
5       0.033    0.007    0.033    0.007        TSPPreparer.py:25(generate_edge_list_tsp)
5       0.026    0.005    0.028    0.006        TSPPreparer.py:98(generate_edge_matrix)
705     0.025    0.000    0.025    0.000        {compile}
5       0.023    0.005    0.078    0.016        binary_quadratic_model.py:595(add_interactions_from)
4       0.020    0.005    0.202    0.050        __init__.py:4(<module>)
11195   0.020    0.000    0.093    0.000        response.py:675(__getitem__)
5       0.016    0.003    0.018    0.004        TSPSolver.py:11(generate_tsp_qubo)
...
```

Listing 2. Output of cProfile for locally used QBSolv.

```
number of clusters: 5

0000000001100000000000000000100100000000000001000000010000000000001000000010000000100000000000000100
-4922.00000 Energy of solution
4 Number of Partitioned calls, 1 output sample
3.31200 seconds of classic cpu time

0010000000000010000000010000000000000000000101000000000000000000010000000000001000000100000000100000000
-7646.00000 Energy of solution
2 Number of Partitioned calls, 1 output sample
2.03200 seconds of classic cpu time

0000000100000000001000000000000010000000010000000000100000000000000001000001000000000000000000100100
-12551.00000 Energy of solution
2 Number of Partitioned calls, 1 output sample
1.25100 seconds of classic cpu time

0001000000001000000000001000000100000000000000010000000000100000001000000000000100000100001000000000
-6903.00000 Energy of solution
4 Number of Partitioned calls, 1 output sample
3.56600 seconds of classic cpu time

0001000000000100000000001000000000000010000000000000100000000000000000010000000000100000000001000000
-16246.00000 Energy of solution
2 Number of Partitioned calls, 1 output sample
1.20000 seconds of classic cpu time

Total distance 566
```

Listing 3. Output of remotely used QBSolv.

```
882498 function calls (821251 primitive calls) in 15.792 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall      filename:lineno(function)
241     11.205   0.046    11.205   0.046        {method 'acquire' of 'thread.lock' objects}
5023    1.028    0.000    1.028    0.000        {built-in method __new__ of type object at 0x53A271E0}
55      0.926    0.017    0.926    0.017        {method 'read' of '_ssl._SSLSocket' objects}
1       0.618    0.618    0.618    0.618        {minorminer.find_embedding}
1       0.368    0.368    0.368    0.368        {method 'do_handshake' of '_ssl._SSLSocket' objects}
1       0.181    0.181    0.181    0.181        {method 'connect' of '_socket.socket' objects}
5       0.077    0.015    11.369   2.274        {dwave_qbsolv.qbsolv_binding.run_qbsolv}
51      0.068    0.001    1.287    0.025        __init__.py:1(<module>)
41476   0.065    0.000    0.065    0.000        binary_quadratic_model.py:484(add_interaction)
5       0.044    0.009    0.045    0.009        TSPPreparer.py:56(generate_adjacency_matrix)
5       0.035    0.007    0.035    0.007        TSPPreparer.py:25(generate_edge_list_tsp)
61672   0.033    0.000    0.049    0.000        records.py:438(__getattribute__)
3       0.031    0.010    0.032    0.011        decoder.py:370(raw_decode)
701     0.026    0.000    0.091    0.000        binary_quadratic_model.py:595(add_interactions_from)
5       0.026    0.005    0.028    0.006        TSPPreparer.py:98(generate_edge_matrix)
705     0.025    0.000    0.025    0.000        {compile}
5       0.017    0.003    0.019    0.004        TSPSolver.py:11(generate_tsp_qubo)
...
```

Listing 4. Output of cProfile for remotely used QBSolv.

```
qpu_access_time 7783            qpu_access_time 7789
qpu_access_time 7787            qpu_access_time 7783
qpu_access_time 7808            qpu_access_time 7807
qpu_access_time 7814            qpu_access_time 7778
qpu_access_time 7768            qpu_access_time 7795
qpu_access_time 7802            qpu_access_time 7788
qpu_access_time 7807            qpu_access_time 7782
```

Listing 5. qpu_access_time of using QBSolv remotely. The qpu_access_time includes the whole time range from programming to post processing overhead time.