qbsolv Documentation

D-Wave Systems Inc

Feb 24, 2020

Contents

A decomposing solver that finds a minimum value of a large quadratic unconstrained binary optimization (QUBO) problem by splitting it into pieces. The pieces are solved using a classical solver running the tabu algorithm. qbsolv also enables configuring a D-Wave system as the solver.

Note: Access to a D-Wave system must be arranged separately.

CHAPTER 1

Example

```
from dwave_qbsolv import QBSolv
Q = {(0, 0): 1, (1, 1): 1, (0, 1): 1}
response = QBSolv().sample_qubo(Q)
print("samples=" + str(list(response.samples())))
print("energies=" + str(list(response.data_vectors['energy'])))
```

CHAPTER 2

Documentation

2.1 Introduction

Divide-and-conquer and dynamic programming algorithms have a rich history in computer science for problems with large numbers of variables. Many hard problems that can benefit from quantum computers are too large to map directly to a QPU. To solve a problem with more variables than the available number of qubits, we break the problem into subproblems, solve the subproblems, and then reconstruct an answer to the original problem from the subproblem solutions.

qbsolv is one such decomposing solver. It provides two interfaces:

- *Command Line Interface* (CLI)

  The tabu algorithm is executed on the problem which is divided into subproblems of several dozen variables each.

- *Python Interface*

  The Python interface provides a `QBSolv` class wrapper for the qbsolv C code. A *dimod* sampler can be substituted for the default tabu algorithm.

For a description of the algorithm and implementation, see `Partitioning Optimization Problems for Hybrid Classical/Quantum Execution`.

For a description of the tabu search algorithm, see Tabu search.

2.1.1 Example

This example sends 30-variable sub-problems of a 500-variable QUBO to the *dwave-neal* sampler to be incorporated into the tabu results run in the main loop of qbsolv.

```
>>> from dwave_qbsolv import QBSolv
>>> import neal
>>> import itertools
>>> import random
```

(continues on next page)

(continued from previous page)

```
...
>>> qubo_size = 500
>>> subqubo_size = 30
>>> Q = {t: random.uniform(-1, 1) for t in itertools.product(range(qubo_size),␣
↪repeat=2)}
>>> sampler = neal.SimulatedAnnealingSampler()
>>> response = QBSolv().sample_qubo(Q, solver=sampler, solver_limit=subqubo_size)
>>> print("energies=" + str(list(response.data_vectors['energy'])))    # doctest: +SKIP
energies=[-2800.794817495185]
```

2.2 Reference Documentation

2.2.1 Command Line Interface

Use the following command with its options to run qbsolv from a terminal.

```
qbsolv -i infile [-o outfile] [-m] [-T] [-n] [-S SubMatrix] [-w]
    [-h] [-a algorithm] [-v verbosityLevel] [-V] [-q] [-t seconds]
```

Description

qbsolv executes a quadratic unconstrained binary optimization (QUBO) problem represented in a file. It returns bit-vector results that minimizes—or optionally, maximizes—the value of the objective function represented by the QUBO. The problem is represented in QUBO(5) file format.

The QUBO input problem is not limited to the graph size or connectivity of a sampler, for example the D-Wave system.

Options are as follows:

```
-i infile
    Name of the file for the input QUBO. This option is mandatory.
-o outfile
    Optional output filename.
    Default is the standard output.
-a algorithm
    Optional selection for the outer loop algorithm.  Default is o.
    'o' for original qbsolv method. Submatrix based upon change in energy.
    'p' for path relinking.  Submatrix based upon differences of solutions
-m
    Optional selection of finding the maximum instead of the minimum.
-T target
    Optional argument target value of the objective function. Stops execution when␣
↪found.
-t timeout
    Optional timeout value. Stops execution when the elapsed CPU time equals or
    exceeds it. Timeout is only checked after completion of the main
    loop. Other halt values such as 'target' and 'repeats' halt before 'timeout'.
    Default value is 2592000.0.
-n repeats
    Optional number of times the main loop of the algorithm is repeated with
    no change in optimal value found before stopping.
    Default value is 50.
-S subproblemSize
```

(continues on next page)

(continued from previous page)

```
    Optional size of the sub-problems into which the QUBO is decomposed.
    If no "-S 0" or "-S" argument is present, uses the size specified in the
    embedding file found in the workspace set up by DW. If no DW environment is
    established, value defaults to 47 and uses the tabu solver on subproblems.
    If a value is specified, subproblems based on that size are solved with the
    tabu solver.
-w
    If present, the QUBO matrix and result are printed in .csv format.
-h
    If present, prints the help or usage message for qbsolv and exits without␣
↪execution.
-v verbosityLevel
    Optional setting of the verbosity of output. The default verbosityLevel of
    0 outputs the number of bits in the solution, the solution,
    and the energy of the solution.  A verbosityLevel of 1 outputs the same
    information for multiple solutions, if found. A verbosityLevel of 2
    also outputs more detailed information at each step of the algorithm. The
    information increases for verbosity levels of up to 4.
-V
    If present, prints the version number of the qbsolv program and exits without␣
↪execution.
-q
    If present, prints the format of the QUBO file.
-r seed
    Used to reset the seed for the random number generation.
```

2.2.2 Python Interface

Class

class QBSolv

Wraps the qbsolv C package for python.

Examples

This example uses the tabu search algorithm to solve a small Ising problem.

```
>>> h = {0: -1, 1: 1, 2: -1}
>>> J = {(0, 1): -1, (1, 2): -1}
>>> response = QBSolv().sample_ising(h, J)
>>> list(response.samples())
'[{0: 1, 1: 1, 2: 1}]'
>>> list(response.energies())
'[1.0]'
```

Methods

QBSolv.sample(bqm, **kwargs)	Sample low-energy states defined by a QUBO using qbsolv.

dwave_qbsolv.QBSolv.sample

QBSolv.**sample**(*bqm*, ***kwargs*)

Sample low-energy states defined by a QUBO using qbsolv.

Note: The qbsolv library being shared by all instances of this class is non-reentrant and not thread safe. The GIL should not be released by this method until that is resolved.

Note: The default build of this library doesn't have the dw library. To use solver='dw' this module must be built from source with that library.

The parameter *solver* given to this method has several valid forms:

- String 'tabu' (default): sub problems are called via an internal call to tabu.
- String 'dw': sub problems are given to the dw library.
- Instance of a dimod sampler. The *sample_qubo* method is invoked.
- Callable that has the signature (qubo: dict, current_best: dict) and returns a result list/dictionary with the new solution.

Parameters

- **Q** (*dict*) – A dictionary defining the QUBO. Should be of the form {(u, v): bias} where u, v are variables and bias is numeric.
- **num_repeats** (*int, optional*) – Determines the number of times to repeat the main loop in qbsolv after determining a better sample. Default 50.
- **seed** (*int, optional*) – Random seed. Default generated by random module.
- **algorithm** (*int, optional*) – Algorithm to use. Default is ENERGY_IMPACT. Algorithm numbers can be imported from the module under the names ENERGY_IMPACT and SOLUTION_DIVERSITY.
- **verbosity** (*int, optional*) – Prints more detail about qbsolv's internal process as this number increases.
- **timeout** (*float, optional*) – Number of seconds before routine halts. Default is 2592000.
- **solver** – Sampling method for qbsolv to use; see method description.
- **solver_limit** (*int, optional*) – Maximum number of variables in a sub problem.
- **target** (*float, optional*) – If given, qbsolv halts when a state with this energy value or better is discoverd. Default is None.
- **find_max** (*bool, optional*) – Switches from searching for minimization to maximization. Default is False (minimization).

Returns Response

Examples

This example uses the tabu search algorithm to solve a small QUBO.

```
>>> Q = {(0, 0): 1, (1, 1): 1, (0, 1): 1}
>>> response = QBSolv().sample_qubo(Q)
>>> list(response.samples())
'[{0: 0, 1: 0}]'
>>> list(response.energies())
'[0.0]'
```

2.2.3 qbsolv Input File Format

A .qubo file contains data that describes an unconstrained quadratic binary optimization problem. It is an ASCII file comprising four types of lines:

1. Comments defined by a "c" in column 1. Comments may appear anywhere in the file, and are ignored.
2. Program line defined by a "p" in the first column. A single program line must be the first non-comment line in the file. The program line has six required fields separated by space(s), as in this example:

```
p   qubo  topology   maxNodes   nNodes   nCouplers
```

   where:

```
p           Problem line sentinel.
qubo        File type identifier.
topology    String that identifies the topology of the problem and the␣
↪specific
            problem type. For an unconstrained problem, target is "0" or
            "unconstrained." In future implementations, valid strings
            might include "chimera128" or "chimera512" (among others).
maxNodes    Number of nodes in the topology.
nNodes      Number of nodes in the problem (nNodes <= maxNodes).
            Each node has a unique number and must take a value in the␣
↪range
            {0 - (maxNodes-1)}. A duplicate node number is an error. Node
            numbers need not be in order, and need not be contiguous.
nCouplers   Number of couplers in the problem. Each coupler is a unique␣
↪connection
            between two different nodes. The maximum number of couplers is␣
↪(nNodes)^2.
            A duplicate coupler is an error.
```

3. nNodes clauses. Each clause is made up of three numbers, separated by one or more blanks. The first two numbers must be integers and are the number for this node (repeated). The node number must be in range {0 , (maxNodes-1)}. The third value is the weight associated with the node. Weight may be an integer or float, and can take on any positive or negative value, or be set to zero.
4. nCouplers clauses. Each clause is made up of three numbers, separated by one or more blanks. The first two numbers, (i and j), are the node numbers for this coupler and must be different integers, where (i < j).Each number must be one of the nNodes valid node numbers (and thus in range {0, (maxNodes-1)}). The third value is the strength associated with the coupler. Strength may be an integer or float, and can take on any positive or negative value, but not zero. Every node must connect with at least one other node (thus must have at least one coupler connected to it).

Here is a simple QUBO file example for an unconstrained QUBO with 4 nodes and 6 couplers. This example is provided to illustrate the elements of a QUBO benchmark file, not to represent a real problem.

```
| <--- column 1
c
c  This is a sample .qubo file
c  with 4 nodes and 6 couplers
c
p  qubo  0  4  4  6
c ------------------
0  0   3.4
1  1   4.5
2  2   2.1
3  3   -2.4
c ------------------
0  1   2.2
0  2   3.4
1  2   4.5
0  3   -2
1  3   4.5678
2  3   -3.22
```

2.3 Installation

2.3.1 Python

A wheel might be available for your system on PyPI. Source distributions are provided as well.

```
pip install dwave-qbsolv
```

Alternatively, you can build the library with setuptools.

```
pip install -r python/requirements.txt
pip install cython==0.27
python setup.py install
```

2.3.2 C

To build the C library use cmake to generate a build command for your system. On Linux the commands would be something like this:

```
mkdir build; cd build
cmake ..
make
```

To build the command line interface turn the cmake option *QBSOLV_BUILD_CMD* on. The command line option for cmake to do this would be *-DQBSOLV_BUILD_CMD=ON*. To build the tests turn the cmake option *QBSOLV_BUILD_TESTS* on. The command line option for cmake to do this would be *-DQBSOLV_BUILD_TESTS=ON*.

2.4 License

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination

of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for

loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

2.5 Contributing to qbsolv

We, the qbsolv developers, expect a variety of contributions, from reporting of bugs or suggestions for examples or tutorials to development of improvements or alternatives to qbsolv's core algorithms. Here we describe how we plan to interact with contributors making such contributions, though that plan will of course change as those contributions occur.

2.5.1 Issues

Bugs or anomalies in the behavior of the tool, issues with its installation, requests for changes in documentation, design and development issues, and feature requests will be tracked via GitHub's Issues mechanism.

When reporting a bug, please provide the following info if appropriate:

- What are the steps to reproduce the bug? If possible, the simplest such set of steps is best.
- Does the bug still happen using the latest version?
- What qbsolv version and OS are you using?

2.5.2 Contributions

We believe that qbsolv will make the fastest progress to being a robust metaheuristic solver capable of quantum acceleration by being widely used both by end-users solving problems and by metaheuristic algorithm developers exploring new metaheuristic algorithms.

In general, you may want to think about starting your contributions gradually, using qbsolv and reporting any strengths and weaknesses (e.g., bugs, documentation improvements) you encounter. This will help you build relationships with the qbsolv developers. And don't forget that contributions can be other than just code or documentation; creating an example or a tutorial helps new users come up to speed and is often high value.

We expect to incorporate promising and proven algorithmic changes into the master code base. This wide use by algorithm developers requires a balance between accepting changes and maintaining a stable tool for end-users. Over time we expect to have processes for both building and regression testing (both correctness and performance) that we will expect new changes to pass before being considered for inclusion.

2.5.3 Submitting A Contribution

For now, we accept a contribution as a Pull Request (PR) on GitHub, though this may change. Please follow these steps:

- If your change is substantial, first create a feature request to start a discussion with the developers to ensure your intent aligns with qbsolv plans.

- A PR should have a clear purpose and do exactly one thing. This enables the rest of the process to be crisp.
- Each commit in PR should be an atomic change representing one step in development.
- As appropriate, please explain anything that is not obvious from the code; this could be in comments, commit messages, or the PR description.
- Sign your patch via the sign-off line, often created by git commit -s. Your sign-off certifies that you wrote the patch or otherwise have the right to pass it along as an open-source patch; see the full text of the certificate just below. Your sign-off line might look like Signed-off-by: Abby Smith <abby.smith@mail.com> Please use your real name, not a pseudonym. This project does not accept anonymous contributions.

2.5.4 License

qbsolv is licensed under the terms in LICENSE. By contributing to the project, you agree to the terms of the license and to release your contribution under those terms.

2.6 Indices and tables

- genindex
- modindex
- search
- Glossary

Python Module Index

d

Index