D-Wave Micro Client dimod Documentation

Release 0.3.0

D-Wave Systems Inc

Feb 09, 2018

Contents

dimod wrapper for the D-Wave Micro Client

CHAPTER 1

Installation

To install:

```
pip install dwave_micro_client_dimod
```

To build from souce:

```
pip install .
```

CHAPTER 2

License

Released under the Apache License 2.0

CHAPTER 3

Documentation

3.1 DWaveSampler

class DWaveSampler(*solver_name=None*, *url=None*, *token=None*, *proxies=None*, *permissive_ssl=False*)

dimod wrapper for a D-Wave Micro Client.

Parameters

- **solver_name** (*str, optional*) – Id of the requested solver. None will return the default solver (see configuration).
- **url** (*str, optional*) – URL of the SAPI server. None will return the default url (see configuration).
- **token** (*str, optional*) – Authentication token from the SAPI server. None will return the default token (see configuration).
- **proxies** (*dict, optional*) – Mapping from the connection scheme (http[s]) to the proxy server address.
- **permissive_ssl** (*boolean, optional, default=False*) – Disables SSL verification.

structure

tuple –

A named 3-tuple with the following properties/values:

nodelist (list): The nodes available to the sampler.

edgelist (list[(node, node)]): The edges available to the sampler.

adjacency (dict): Encodes the edges of the sampler in nested dicts. The keys of adjacency are the nodes of the sampler and the values are neighbor-dicts.

accepted_kwargs
dict[str, dimod.SamplerKeywordArg] – The keyword arguments accepted by the *sample_ising* and *sample_qubo* methods for this sampler.

my_kwargs()
The keyword arguments accepted by DWaveSampler

Returns SamplerKeywordArg]: The keyword arguments accepted by the *sample_ising* and *sample_qubo* methods for this sampler or the top-level composite layer. For all accepted keyword arguments see *accepted_kwargs*.

Return type dict[str

sample_ising(*linear*, *quadratic*, ***kwargs*)
Sample from the provided Ising model.

Parameters

- **linear** (*list/dict*) – Linear terms of the model.
- **quadratic** (*dict of (int, int)* – float): Quadratic terms of the model.
- ****kwargs** – Parameters for the sampling method, specified per solver.

Returns *FutureResponse*

sample_qubo(*Q*, ***kwargs*)
Sample from the provided QUBO.

Parameters

- **Q** (*dict*) – The coefficients of the QUBO.
- ****kwargs** – Parameters for the sampling method, specified per solver.

Returns *FutureResponse*

3.2 EmbeddingComposite

class EmbeddingComposite(*sampler*)
Composite to map unstructured problems to a structured sampler.

Parameters **sampler** (dimod.TemplateSampler) – A structured dimod sampler to be wrapped.

sample_ising(*h*, *J*, ***kwargs*)
Sample from the provided unstructured Ising model.

Parameters

- **h** (*list/dict*) – Linear terms of the model.
- **J** (*dict of (int, int)* – float): Quadratic terms of the model.
- ****kwargs** – Parameters for the sampling method, specified per solver.

Returns dimod.SpinResponse

accepted_kwargs
dict[str – SamplerKeywordArg] – The keyword arguments accepted by the *sample_ising* and *sample_qubo* methods for this sampler.

my_kwargs()
The keyword arguments accepted by the sampler or the highest composite layer.

Returns SamplerKeywordArg]: The keyword arguments accepted by the *sample_ising* and *sample_qubo* methods for this sampler or the top-level composite layer. For all accepted keyword arguments see *accepted_kwargs*.

Return type dict[str

Note: This method is inherited from the TemplateSampler base class.

---

sample_qubo(*Q*, ***kwargs*)

Converts the given QUBO into an Ising problem, then invokes the sample_ising method.

See sample_ising documentation for more information.

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.
- ****kwargs** – Any keyword arguments are passed directly to sample_ising.

Returns A *BinaryResponse*, converted from the *SpinResponse* return from sample_ising.

Return type BinaryResponse

Note: This method is inherited from the TemplateSampler base class.

---

structure

Structure for the sampler. None for unstructured samplers.

3.3 TilingComposite

class TilingComposite(*sampler*, *sub_m*, *sub_n*, *t=4*)

Composite to tile a small problem across a Chimera-structured sampler. A problem that can fit on a small Chimera graph can be replicated across a larger Chimera graph to get samples from multiple areas of the system in one call. For example, a 2x2 Chimera lattice could be tiled 64 times (8x8) on a fully-yielded D-WAVE 2000Q system (16x16).

Parameters

- **sampler** (dimod.TemplateSampler) – A structured dimod sampler to be wrapped.
- **sub_m** (*int*) – The number of rows in the sub Chimera lattice.
- **sub_n** (*int*) – The number of columns in the sub Chimera lattice.
- **t** (*int*) – The size of the shore within each Chimera cell.

structure

tuple –

A named 3-tuple with the following properties/values:

nodelist (list): The nodes available to the sampler.

edgelist (list[(node, node)]): The edges available to the sampler.

adjacency (dict): Encodes the edges of the sampler in nested dicts. The keys of adjacency are the nodes of the sampler and the values are neighbor-dicts.

embeddings
list – A list of dictionaries mapping from the sub Chimera lattice to the structured sampler of the form {v: {s, ...}, ...} where v is a variable in the sub Chimera lattice and s is a variable in the system.

sample_ising(*h*, *J*, ***kwargs*)
Sample from the sub Chimera lattice.

Parameters

- **h** (*list/dict*) – Linear terms of the model.
- **J** (*dict of (int, int)* – float): Quadratic terms of the model.
- ****kwargs** – Parameters for the sampling method, specified per solver.

Returns `dimod.SpinResponse`

draw_tiling(*sampler*, *t=4*)
Draw Chimera graph of sampler with colored tiles.

Parameters

- **sampler** (`dwave_micro_client_dimod.TilingComposite`) – A tiled dimod sampler to be drawn.
- **t** (*int*) – The size of the shore within each Chimera cell.

Uses *dwave_networkx.draw_chimera* (see draw_chimera). Linear biases are overloaded to color the graph according to which tile each Chimera cell belongs to.

3.4 FutureResponse

class FutureResponse(*info=None*, *vartype=None*)
A response object for async samples added to it.

Parameters

- **info** (*dict*) – Information about the response as a whole.
- **vartype** (`dimod.Vartype`) – The values that the variables in each sample can take. See `dimod.Vartype`.

add_samples_future(*future*)
Add samples from a micro client Future.

Parameters future (`dwave_micro_client.Future`) – A Future from the dwave_micro_client.

datalist
list – The data in order of insertion. Each datum in data is a dict containing 'sample', 'energy', and 'num_occurences' keys as well an any other information added on insert. This attribute should be treated as read-only, as changing it can break the response's internal logic.

add_data_from(*data*)
Loads data into the response.

Parameters data (*iterable[dict]*) – An iterable of datum. Each datum is a dict. The datum must contain 'sample' and 'energy' keys with dict and number values respectively.

Examples

```
>>> response = dimod.TemplateResponse
>>> response.add_data_from([{'energy': -1, 'num_occurences': 1, 'sample': {0:␣
↪1}},
                            {'energy': 1, 'num_occurences': 1, 'sample': {0: -
↪1}}])
```

add_sample(*sample*, *energy*, *num_occurences=1*, ***kwargs*)
Loads a sample and associated energy into the response.

Parameters

- **sample** (*dict*) – A sample as would be returned by a discrete model solver. Should be a dict of the form {var: value, ... }.
- **energy** (*number*) – The energy associated with the given sample.
- **num_occurences** (*int*) – The number of times the sample occurred.
- ****kwargs** –

Examples

```
>>> response = dimod.TemplateResponse()
>>> response.add_sample({0: -1}, 1)
>>> response.add_sample({0: 1}, -1, sample_idx=1)
>>> list(response.data())
[{'energy': -1, 'num_occurences': 1, 'sample': {0: 1}, 'sample_idx': 1},
 {'energy': 1, 'num_occurences': 1, 'sample': {0: -1}}]
```

add_samples_from(*samples*, *energies*, *num_occurences=1*, ***kwargs*)
Loads samples and associated energies from iterators.

Parameters

- **samples** (*iterator*) – An iterable object that yields samples. Each sample should be a dict of the form {var: value, ... }.
- **energies** (*iterator*) – An iterable object that yields energies associated with each sample.
- **num_occurences** (*int*) – Default 1. The number of times the sample occurred. This is applied to each sample.
- ****kwargs** –

Examples

```
>>> response = dimod.TemplateResponse()
>>> response.add_samples_from([{0: -1}, {0: 1}], [1, -1], dataval='test')
>>> list(response.data())
[{'dataval': 'test', 'energy': -1, 'num_occurences': 1, 'sample': {0: 1}},
 {'dataval': 'test', 'energy': 1, 'num_occurences': 1, 'sample': {0: -1}}]
```

cast(*response_class*, *varmap=None*, *offset=0.0*)
Casts the response to a different type of dimod response.

Parameters

- **response_class** (*type*) – A dimod response class.
- **varmap** (*dict, optional*) – A dict mapping a change in sample values. If not provided samples are not changed.
- **offset** (*number, optional*) – Default 0.0. The energy offset to apply to all of the energies in the response.

Returns A dimod response.

Return type response_class

data(*keys=None*, *ordered_by_energy=True*)

An iterator over the data.

Parameters

- **keys** (*list, optional*) – are provided, data yields a tuple of the values associated with each key in each datum.
- **ordered_by_energy** (*bool, optional*) – Default True. If True, the datum (or tuples) are yielded in order energy, low-to-high. If False, they are yielded in order of insertion.

Yields *dict* – The datum stored in response.

If keys are provided, returns a tuple (see parameter description above and example below).

Examples

```
>>> response = dimod.TemplateResponse()
>>> response.add_samples_from([{0: -1}, {0: 1}], [1, -1])
>>> list(response.data())
[{'energy': -1, 'num_occurences': 1, 'sample': {0: 1}},
 {'energy': 1, 'num_occurences': 1, 'sample': {0: -1}}]
>>> for sample in response.data(keys=['sample']):
...     pass
>>> for sample, num_occurences in response.data(keys=['sample', 'num_
↪occurences']):
...     pass
```

done()

True if all of the futures added to the response have arrived.

energies(*data=False*)

An iterator over the energies.

Parameters **data** (*bool, optional*) – Default False. If True, returns an iterator of 2-tuples, (sample, datum).

Yields *number* – The energies, from low-to-high.

If data=True, yields 2-tuple (energy, datum). Where datum is the data associated with the given energy.

items(*data=False*)

Iterator over the samples and energies.

Parameters **data** (*bool, optional*) – If True, return an iterator of 3-tuples (sample, energy, data). If False return an iterator of 2-tuples (sample, energy) over all of the samples and energies. Default False.

Returns If data is False, return an iterator of 2-tuples (sample, energy) over all samples and energies in response in order of increasing energy. If data is True, return an iterator of 3-tuples (sample, energy, data) in order of increasing energy.

Return type iterator

Examples

```
>>> response = TemplateResponse()
>>> response.add_sample({0: -1}, 1, data={'n': 5})
>>> response.add_sample({0: 1}, -1, data={'n': 1})
>>> list(response.items())
[({0: 1}, -1), ({0: -1}, 1)]
>>> list(response.items(data=True))
[({0: 1}, -1, {'n': 1}), ({0: -1}, 1, {'n': 5})]
```

Note: Depreciation Warning: This method of access is being depreciated. it can be replaced by *response.data(keys=['sample', 'energy'])*

---

relabel_samples(*mapping*)

Return a new response object with the samples relabeled.

Parameters **mapping** (*dict[hashable, hashable]*) – A dictionary with the old labels as keys and the new labels as values. A partial mapping is allowed.

Examples

```
>>> response = TemplateResponse()
>>> response.add_sample({'a': -1, 'b': 1}, 1)
>>> response.add_sample({'a': 1, 'b': -1}, -1)
>>> mapping = {'a': 1, 'b': 0}
```

```
>>> new_response = response.relabel_samples(mapping)
>>> list(new_response.samples())
[{0: -1, 1: 1}, {0: 1, 1: -1}]
```

samples(*data=False*)

An iterator over the samples.

Parameters **data** (*bool, optional*) – Default False. If True, returns an iterator of 2-tuples, (sample, datum).

Yields *dict* – The samples, in order of energy low-to-high.

If data=True, yields 2-tuple (sample, datum). Where datum is the data associated with the given sample.

Examples

```
>>> response = TemplateResponse()
>>> response.add_samples_from([{0: -1}, {0: 1}], [1, -1])
>>> list(response.samples())
[{0: 1}, {0: -1}]
```

sorted_datalist
list[dict] – The data in order of energy, low-to-high. The datum stored in sorted_datalist are the same as in datalist. This list is generated on the first read after an insertion to the response.

3.5 License

CHAPTER 4

Indices and tables

- genindex
- modindex
- search
- Glossary

Python Module Index

Index