D-Wave Embedding Utilities Documentation

Release 0.3.0

D-Wave Systems Inc

Feb 21, 2018

Contents

This package provides functions that map samples between a source graph and a target graph.

CHAPTER 1

Terminology

model - A collection of variables with associated linear and quadratic biases. Sometimes referred to in other projects as a **problem**. In this project all models are expected to be spin-valued - that is the variables in the model can be -1 or 1.

graph - A collection of nodes and edges. A graph can be derived from a model; a node for each variable and an edge for each pair of variables with a non-zero quadratic bias.

source - The model or induced graph that we wish to embed. Sometimes referred to in other projects as the **logical** graph/model.

target - Embedding attempts to create a target model from a target graph. The process of embedding takes a source model, derives the source graph, maps the source graph to the target graph, then derives the target model. Sometimes referred to in other projects at the **embedded** graph/model.

chain - A collection of nodes or variables in the target graph/model that we want to act like a single node/variable.

chain strength - The magnitude of the negative quadratic bias applied between variables within a chain.

Examples

Imagine that we have a sampler which is structured as a 4-cycle graph.

```
import networkx as nx
target_graph = nx.cycle_graph(4)
# target_graph = {0: {1, 3}, 1: {0, 2}, 2: {1, 3}, 3: {0, 2}}  # equivalent
```

We have a model on a 3-cycle that we wish to embed.

```
source_linear = {'a': 0., 'b': 0., 'c': 0.}
source_quadratic = {('a', 'b'): 1., ('b', 'c'): 1., ('a', 'c'): 1.}
```

Finally, we have an embedding that maps a 3-cycle to a 4-cycle. In this case we want variables 1, 2 in the target to behave as a single variable.

```
embedding = {'a': {0}, 'b': {1, 2}, 'c': {3}}
```

To get the target model, use the *embed_ising()* function.

```
target_linear, target_quadratic, chain_quadratic = embed_ising(
    source_linear, source_quadratic, embedding, target_graph)
```

Say that we sample from the target model using some sampler, we can then umembed the samples using *unembed_samples()*.

```
samples = {0: -1, 1: -1, 2: 1, 3: 1}
source_samples = unembed_samples(samples, embedding)
```

CHAPTER 2

Functions

embed_ising(*source_linear*, *source_quadratic*, *embedding*, *target_adjacency*, *chain_strength=1.0*)

Embeds a logical Ising model onto another graph via an embedding.

Parameters

- **source_linear** (*dict*) – The linear biases to be embedded. Should be a dict of the form {v: bias, ...} where v is a variable in the source model and bias is the linear bias associated with v.
- **source_quadratic** (*dict*) – The quadratic biases to be embedded. Should be a dict of the form {(u, v): bias, ...} where u, v are variables in the source model and bias is the quadratic bias associated with (u, v).
- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a variable in the source model and s is a variable in the target model.
- **target_adjacency** (dict/`networkx.Graph`) – The adjacency dict of the target graph. Should be a dict of the form {s: Ns, ...} where s is a variable in the target graph and Ns is the set of neighbours of s.
- **chain_strength** (*float, optional*) – The quadratic bias that should be used to create chains.

Returns

A 3-tuple containing:

dict: The linear biases of the target problem. In the form {s: bias, ...} where s is a node in the target graph and bias is the associated linear bias.

dict: The quadratic biases of the target problem. A dict of the form {(s, t): bias, ...} where (s, t) is an edge in the target graph and bias is the associated quadratic bias.

dict: The quadratic biases that induce the variables in the target problem to act as one. A dict of the form {(s, t): -chain_strength, ...} which is the quadratic biases associated with the chains.

Return type (dict, dict, dict)

Examples

```
>>> source_linear = {'a': 1, 'b': 1}
>>> source_quadratic = {('a', 'b'): -1}
>>> embedding = {'a': [0, 1], 'b': [2]}
>>> target_adjacency = {0: {1, 2}, 1: {0, 2}, 2: {0, 1}}
>>> target_linear, target_quadratic, chain_quadratic = embed_ising(
...     source_linear, source_quadratic, embedding, target_adjacency)
>>> target_linear
{0: 0.5, 1: 0.5, 2: 1.0}
>>> target_quadratic
{(0, 2): -0.5, (1, 2): -0.5}
>>> chain_quadratic
{(0, 1): -1.0}
```

target_to_source(*target_adjacency*, *embedding*)

Derive the source adjacency from an embedding and target adjacency.

Parameters

- **target_adjacency** (dict/networkx.Graph) – A dict of the form {v: Nv, ...} where v is a node in the target graph and Nv is the neighbors of v as an iterable. This can also be a networkx graph.
- **embedding** (*dict*) – A mapping from a source graph to a target graph.

Returns The adjacency of the source graph.

Return type dict

Raises ValueError – If any node in the target_adjacency is assigned more than one node in the source graph by embedding.

chain_break_frequency(*samples*, *embedding*)

Determines the frequency of chain breaks in the given samples.

Parameters

- **samples** (*iterable*) – An iterable of samples where each sample is a dict of the form {v: val, ...} where v is a variable in the target graph and val is the associated value as determined by a binary quadratic model sampler.
- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a variable in the source model and s is a variable in the target model.

Returns The frequency of chain breaks in the form {v: f, ...} where v is a variable in the source graph and frequency is the fraction of chains that were broken as a float.

Return type dict

edgelist_to_adjacency(*edgelist*)

Converts an iterator of edges to an adjacency dict.

Parameters **edgelist** (*iterable*) – An iterator over 2-tuples where each 2-tuple is an edge.

Returns The adjacency dict. A dict of the form {v: Nv, ...} where v is a node in a graph and Nv is the neighbors of v as an set.

Return type dict

chain_to_quadratic(*chain*, *target_adjacency*, *chain_strength*)
Determine the quadratic biases that induce the given chain.

Parameters

- **chain** (*set/list/tuple*) – The variables that make up a chain.
- **target_adjacency** (dict/networkx.Graph) – The adjacency dict of the target graph. Should be a dict of the form {s: Ns, ...} where s is a variable in the target graph and Ns is the set of neighbours of s.
- **chain_strength** (*float*) – The quadratic bias that should be used to create chains.

Returns The quadratic biases that induce the given chain.

Return type dict[edge, float]

Examples

```
>>> chain = {1, 2}
>>> target_adjacency = {0: {1, 2}, 1: {0, 2}, 2: {0, 1}}
>>> chain_to_quadratic(chain, target_adjacency, 1)
{(1, 2): -1}
```

unembed_samples(*samples*, *embedding*, *chain_break_method=None*)
Return samples over the variables in the source graph.

Parameters

- **samples** (*iterable*) – An iterable of samples where each sample is a dict of the form {v: val, ...} where v is a variable in the target model and val is the associated value as determined by a binary quadratic model sampler.
- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a node in the source graph and s is a node in the target graph.
- **chain_break_method** (*function, optional*) – The method used to resolve chain breaks. Default is **:method:'majority_vote'**.

Returns A list of unembedded samples. Each sample is a dict of the form {v: val, ...} where v is a variable in the source graph and val is the value associated with the variable.

Return type list

CHAPTER 3

Chain Break Methods

discard(*sample*, *embedding*)
Discards the sample if broken.

Parameters

- **sample** (*dict*) – A sample of the form {v: val, ...} where v is a variable in the target graph and val is the associated value as determined by a binary quadratic model sampler.
- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a node in the source graph and s is a node in the target graph.

Yields *dict* – The unembedded sample is no chains were broken.

majority_vote(*sample*, *embedding*)
Determines the sample values by majority vote.

Parameters

- **sample** (*dict*) – A sample of the form {v: val, ...} where v is a variable in the target graph and val is the associated value as determined by a binary quadratic model sampler.
- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a node in the source graph and s is a node in the target graph.

Yields *dict* – The unembedded sample. When there is a chain break, the value is chosen to match the most common value in the chain.

weighted_random(*sample*, *embedding*)
Determines the sample values by weighed random choice.

Parameters

- **sample** (*dict*) – A sample of the form {v: val, ...} where v is a variable in the target graph and val is the associated value as determined by a binary quadratic model sampler.

- **embedding** (*dict*) – The mapping from the source graph to the target graph. Should be of the form {v: {s, ...}, ...} where v is a node in the source graph and s is a node in the target graph.

Yields *dict* – The unembedded sample. When there is a chain break, the value is chosen randomly, weighted by the frequency of the values within the chain.

Python Module Index

d

Index