penaltymodel Documentation

D-Wave Systems Inc

Feb 24, 2020

Contents

One approach to solve a constraint satisfaction problem (CSP) using an Ising model or a QUBO, is to map each individual constraint in the CSP to a 'small' Ising model or QUBO. This mapping is called a *penalty model*.

Imagine that we want to map an AND clause to a QUBO. In other words, we want the solutions to the QUBO (the solutions that minimize the energy) to be exactly the valid configurations of an AND gate. Let $z = AND(x_1, x_2)$.

Before anything else, let's import that package we will need.

```
import penaltymodel.core as pm
import dimod
import networkx as nx
```

Next, we need to determine the feasible configurations that we wish to target (by making the energy of these configuration in the binary quadratic low). Below is the truth table representing an AND clause.

Table 1: AND Gate

x_1	x_2	z
0	0	0
0	1	0
1	0	0
1	1	1

The rows of the truth table are exactly the feasible configurations.

```
feasible_configurations = {(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 1)}
```

We also need a target graph and to label the decision variables. We create a node in the graph for each variable in the problem, and we add an edge between each node, represnting the interactions between the variables. In this case we allow an interaction between every variable, but more sparse interactions are possible. The labels of the nodes and the decision variables match.

```
graph = nx.Graph()
graph.add_edges_from([('x1', 'x2'), ('x1', 'z'), ('x2', 'z')])
decision_variables = ['x1', 'x2', 'z']
```

We now have everything needed to build our Specification. We have binary variables so we select the appropriate variable type.

```
spec = pm.Specification(graph, decision_variables, feasible_configurations, dimod.
↪BINARY)
```

At this point, if we have any factories installed, we could use the factory interface to get an appropriate penalty model for our specification.

```
p_model = pm.get_penalty_model(spec)
```

However, if we know the QUBO, we can build the penalty model ourselves. We observe that for the equation:

$$E(x_1, x_2, z) = x_1 x_2 - 2(x_1 + x_2)z + 3z + 0$$

We get the following energies for each row in our truth table.

x_1	x_2	z	$E(x_1, x_2, z)$
0	0	0	0
0	0	1	3
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

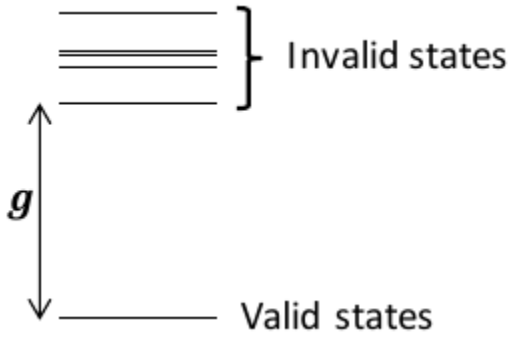
We can see that the energy is minimized on exactly the desired feasible configurations. So we encode this energy function as a QUBO. We make the offset 0.0 because there is no constant energy offset.

```
qubo = dimod.BinaryQuadraticModel({'x1': 0., 'x2': 0., 'z': 3.},
                                  {('x1', 'x2'): 1., ('x1', 'z'): 2., ('x2', 'z'): 2.},
                                  0.0,
                                  dimod.BINARY)
```

We know from the table that our ground energy is 0, but we can calculate it using the qubo to check that this is true for the feasible configuration $(0, 1, 0)$.

```
ground_energy = qubo.energy({'x1': 0, 'x2': 1, 'z': 0})
```

The last value that we need is the classical gap. This is the difference in energy between the lowest infeasible state and the ground state.



With all of the pieces, we can now build the penalty model.

```
classical_gap = 1
p_model = pm.PenaltyModel.from_specification(spec, qubo, classical_gap, ground_energy)
```

CHAPTER 1

Documentation

Note: This documentation is for the latest version of penaltymodel. Documentation for the version currently installed by dwave-ocean-sdk is here: penaltymodel.

1.1 Packages

Release

Date Feb 24, 2020

1.1.1 penaltymodel

The core penalty model package. Contains the logic shared in the penalty model ecosystem.

Included Classes

PenaltyModel

class PenaltyModel(*graph*, *decision_variables*, *feasible_configurations*, *vartype*, *model*, *classical_gap*, *ground_energy*, *ising_linear_ranges=None*, *ising_quadratic_ranges=None*)

Container class for the components that make up a penalty model.

A penalty model is a small Ising problem or QUBO that has ground states that match the feasible configurations and excited states that have a classical energy greater than the ground energy by at least the classical gap.

PenaltyModel is a subclass of *Specification*.

Parameters

- **graph** (networkx.Graph/iterable[edge]) – Defines the structure of the desired binary quadratic model. Each node in the graph represents a variable and each edge defines an interaction between two variables. If given as an iterable of edges, the graph will be constructed by adding each edge to an (initially) empty graph.
- **decision_variables** (*iterable*) – The labels of the penalty model's decision variables. Each variable label in *decision_variables* must correspond to a node in *graph*. Should be an ordered iterable of hashable labels.
- **feasible_configurations** (*dict[tuple[int], number]/ iterable[tuple[int]]*) – The set of feasible configurations. Defines the allowed configurations of the decision variables allowed by the constraint. Each feasible configuration should be a tuple, each element of which must be of a value matching *vartype*. If given as a dict, the key is the feasible configuration and the value is the desired relative energy. If given as an iterable, it will be case to a dict where the relative energies are all 0.
- **vartype** (Vartype/str/set) – The variable type desired for the penalty model. Accepted input values: Vartype.SPIN, 'SPIN', {-1, 1} Vartype.BINARY, 'BINARY', {0, 1}
- **model** (dimod.BinaryQuadraticModel) – A binary quadratic model that has ground states that match the feasible_configurations.
- **classical_gap** (*numeric*) – The difference in classical energy between the ground state and the first excited state. Must be positive.
- **ground_energy** (*numeric*) – The minimum energy of all possible configurations.
- **ising_linear_ranges** (*dict[node, [number, number]], optional, default=None*) – When the penalty model is spin-valued, specifies the allowed range for each of the linear biases. If a dict, should be of the form {v: [min, max], ...} where v is a variable in the desired penalty model and (min, max) defines the acceptable range for the linear bias associated with v. If None, the default will be set to {v: [-1, 1], ...} for each v in graph. A partial assignment is allowed.
- **ising_quadratic_ranges** (*dict[node, dict[node, [number, number]], optional, default=None*) – When the penalty model is spin-valued, specifies the allowed range for each of the quadratic biases. If a dict, should be of the form {v: {u: [min, max], ...}, ...} where u and v are variables in the desired penalty model and u, v have an interaction - there is an edge between nodes u, v in *graph*. (min, max) the acceptable range for the quadratic bias associated with u, v. If None, the default will be set to {v: {u: [min, max], ...}, u: {v: [min, max], ...}, ...} for each edge u, v in graph. A partial assignment is allowed.

Examples

The penalty model can be created from its component parts:

```
>>> import networkx as nx
>>> import dimod
>>> graph = nx.path_graph(3)
>>> decision_variables = (0, 2)  # the ends of the path
>>> feasible_configurations = {(-1, -1), (1, 1)}  # we want the ends of the path␣
↪to agree
>>> model = dimod.BinaryQuadraticModel({0: 0, 1: 0, 2: 0}, {(0, 1): -1, (1, 2): -
↪1}, 0.0, dimod.SPIN)
>>> classical_gap = 2.0
```

(continues on next page)

(continued from previous page)

```
>>> ground_energy = -2.0
>>> widget = pm.PenaltyModel(graph, decision_variables, feasible_configurations,␣
↪dimod.SPIN,
...                           model, classical_gap, ground_energy)
```

Or it can be created from a specification:

```
>>> spec = pm.Specification(graph, decision_variables, feasible_configurations,␣
↪dimod.SPIN)
>>> widget = pm.PenaltyModel.from_specification(spec, model, classical_gap,␣
↪ground_energy)
```

decision_variables
Maps the feasible configurations to the graph.

Type tuple

classical_gap
The difference in classical energy between the ground state and the first excited state. Must be positive.

Type numeric

feasible_configurations
The set of feasible configurations. The value is the (relative) energy of each of the feasible configurations.

Type dict[tuple[int], number]

graph
The graph that defines the relation between variables in the penaltymodel. The node labels will be used as the variable labels in the binary quadratic model.

Type `networkx.Graph`

ground_energy
The minimum energy of all possible configurations.

Type numeric

ising_linear_ranges
Defines the energy ranges available for the linear biases of the penalty model.

Type dict[node, (number, number)]

model
A binary quadratic model that has ground states that match the feasible_configurations.

Type `dimod.BinaryQuadraticModel`

ising_quadratic_ranges
Defines the energy ranges available for the quadratic biases of the penalty model.

Type dict[edge, (number, number)]

vartype
The variable type.

Type `dimod.Vartype`

classmethod from_specification(*specification*, *model*, *classical_gap*, *ground_energy*)
Construct a PenaltyModel from a Specification.

Parameters

- **specification** (*Specification*) – A specification that was used to generate the model.
- **model** (dimod.BinaryQuadraticModel) – A binary quadratic model that has ground states that match the feasible_configurations.
- **classical_gap** (*numeric*) – The difference in classical energy between the ground state and the first excited state. Must be positive.
- **ground_energy** (*numeric*) – The minimum energy of all possible configurations.

Returns *PenaltyModel*

relabel_variables(*mapping*, *inplace=True*)

Relabel the variables and nodes according to the given mapping.

Parameters

- **mapping** (*dict[hashable, hashable]*) – A dict with the current variable labels as keys and new labels as values. A partial mapping is allowed.
- **inplace** (*bool, optional, default=True*) – If True, the penalty model is updated in-place; otherwise, a new penalty model is returned.

Returns A PenaltyModel with the variables relabeled according to mapping.

Return type *PenaltyModel*

Examples

```
>>> spec = pm.Specification(nx.path_graph(3), (0, 2), {(-1, -1), (1, 1)},␣
↪dimod.SPIN)
>>> model = dimod.BinaryQuadraticModel({0: 0, 1: 0, 2: 0}, {(0, 1): -1, (1,␣
↪2): -1}, 0.0, dimod.SPIN)
>>> penalty_model = pm.PenaltyModel.from_specification(spec, model, 2., -2.)
>>> relabeled_penalty_model = penalty_model.relabel_variables({0: 'a'},␣
↪inplace=False)
>>> relabeled_penalty_model.decision_variables
('a', 2)
```

```
>>> spec = pm.Specification(nx.path_graph(3), (0, 2), {(-1, -1), (1, 1)},␣
↪dimod.SPIN)
>>> model = dimod.BinaryQuadraticModel({0: 0, 1: 0, 2: 0}, {(0, 1): -1, (1,␣
↪2): -1}, 0.0, dimod.SPIN)
>>> penalty_model = pm.PenaltyModel.from_specification(spec, model, 2., -2.)
>>> __ = penalty_model.relabel_variables({0: 'a'}, inplace=True)
>>> penalty_model.decision_variables
('a', 2)
```

Specification

class Specification(*graph*, *decision_variables*, *feasible_configurations*, *vartype*, *ising_linear_ranges=None*, *ising_quadratic_ranges=None*, *min_classical_gap=2*)

Specification for a PenaltyModel.

See *PenaltyModel* documentation for a fuller description of the different components. A specification can be thought of as an incomplete penalty model.

Parameters

- **graph** (networkx.Graph/iterable[edge]) – Defines the structure of the desired binary quadratic model. Each node in the graph represents a variable and each edge defines an interaction between two variables. If given as an iterable of edges, the graph will be constructed by adding each edge to an (initially) empty graph.
- **decision_variables** (*iterable*) – The labels of the penalty model's decision variables. Each variable label in *decision_variables* must correspond to a node in *graph*. Should be an ordered iterable of hashable labels.
- **feasible_configurations** (*dict[tuple[int], number]/ iterable[tuple[int]]*) – The set of feasible configurations. Defines the allowed configurations of the decision variables allowed by the constraint. Each feasible configuration should be a tuple, each element of which must be of a value matching *vartype*. If given as a dict, the key is the feasible configuration and the value is the desired relative energy. If given as an iterable, it will be case to a dict where the relative energies are all 0.
- **vartype** (dimod.Vartype/str/set) – The variable type desired for the penalty model. Accepted input values: `Vartype.SPIN, 'SPIN', {-1, 1}` `Vartype.BINARY, 'BINARY', {0, 1}`
- **ising_linear_ranges** (*dict[node, [number, number]], optional, default=None*) – When the penalty model is spin-valued, specifies the allowed range for each of the linear biases. If a dict, should be of the form {v: [min, max], ...} where v is a variable in the desired penalty model and (min, max) defines the acceptable range for the linear bias associated with v. If None, the default will be set to {v: [-1, 1], ...} for each v in graph. A partial assignment is allowed.
- **ising_quadratic_ranges** (*dict[node, dict[node, [number, number]], optional, default=None*) – When the penalty model is spin-valued, specifies the allowed range for each of the quadratic biases. If a dict, should be of the form {v: {u: [min, max], ...}, ...} where u and v are variables in the desired penalty model and u, v have an interaction - there is an edge between nodes u, v in *graph*. (min, max) the acceptable range for the quadratic bias associated with u, v. If None, the default will be set to {v: {u: [min, max], ...}, u: {v: [min, max], ...}, ...} for each edge u, v in graph. A partial assignment is allowed.

Examples

```
>>> import networkx as nx
>>> import dimod
>>> graph = nx.path_graph(5)
>>> decision_variables = (0, 4)  # the ends of the path
>>> feasible_configurations = {(-1, -1), (1, 1)}  # we want the ends of the path␣
↪to agree
>>> vartype = dimod.Vartype.SPIN
>>> spec = pm.Specification(graph, decision_variables, feasible_configurations,␣
↪vartype)
```

If we want to make the interaction between (0, 1) ferromagnetic (negative):

```
>>> ising_quadratic_ranges = {0: {1: (-1, 0)}}
>>> spec = pm.Specification(graph, decision_variables, feasible_configurations,␣
↪vartype)
```

decision_variables
The labels of the penalty model's decision variables. Each variable label in *decision_variables* must correspond to a node in *graph*.

Type tuple

feasible_configurations
The set of feasible configurations. Defines the allowed configurations of the decision variables allowed by the constraint. The key is the allowed configuration, the value is the relative energy of each configuration.

Type dict[tuple[int], number]

graph
Defines the structure of the desired binary quadratic model. Each node in the graph represents a variable and each edge defines an interaction between two variables.

Type `networkx.Graph`

ising_linear_ranges
When the penalty model is spin-valued, specifies the allowed range for each of the linear biases. A dict of the form {v: [min, max], ...} where v is a variable in the desired penalty model and [min, max] defines the acceptable range for the linear bias associated with v.

Type dict[node, [number, number], optional, default=None

ising_quadratic_ranges
When the penalty model is spin-valued, specifies the allowed range for each of the quadratic biases. A dict of the form {v: {u: [min, max], ...}, u: {v: [min, max], ...}, ...} where u and v are variables in the desired penalty model and u, v have an interaction - there is an edge between nodes u, v in *graph*.

Type dict[node, dict[node, [number, number]]], optional, default=None

min_classical_gap
This is a threshold value for the classical gap. It describes the minimum energy gap between the highest feasible state and the lowest infeasible state. Default value is 2.

Type float

relabel_variables(*mapping*, *inplace=True*)
Relabel the variables and nodes according to the given mapping.

Parameters

- **mapping** (*dict*) – a dict mapping the current variable/node labels to new ones.
- **inplace** (*bool, optional, default=True*) – If True, the specification is updated in-place; otherwise, a new specification is returned.

Returns A Specification with the variables relabeled according to mapping. If copy=False returns itself, if copy=True returns a new Specification.

Return type *Specification*

Using PenaltyModel Factories

penaltymodel provides functionality for accessing PenaltyModel factories.

Accessing Factories

Any factories that have been identified through the *FACTORY_ENTRYPOINT* entrypoint and installed on the python path can be accessed through the *get_penalty_model()* function.

Examples

```
>>> import networkx as nx
>>> import dimod
>>> graph = nx.path_graph(5)
>>> decision_variables = (0, 4)  # the ends of the path
>>> feasible_configurations = {(-1, -1), (1, 1)}  # we want the ends of the path to␣
↪agree
>>> spec = pm.Specification(graph, decision_variables, feasible_configurations, dimod.
↪SPIN)
>>> widget = pm.get_penalty_model(spec)
```

Functions and Utilities

FACTORY_ENTRYPOINT = 'penaltymodel_factory'
constant used when assigning entrypoints for factories.

Type str

CACHE_ENTRYPOINT = 'penaltymodel_cache'
constant used when assigning entrypoints for caches.

Type str

get_penalty_model(*specification*)
Retrieve a PenaltyModel from one of the available factories.

Parameters **specification** (*Specification*) – The specification for the desired PenaltyModel.

Returns A PenaltyModel as returned by the highest priority factory, or None if no factory could produce it.

Return type *PenaltyModel*/None

Raises ImpossiblePenaltyModel – If the specification describes a penalty model that cannot be built by any factory.

penaltymodel_factory(*priority*)
Decorator to assign a *priority* attribute to the decorated function.

Parameters **priority** (*int*) – The priority of the factory. Factories are queried in order of decreasing priority.

Examples

Decorate penalty model factories like:

```
>>> @pm.penaltymodel_factory(105)
... def factory_function(spec):
...     pass
>>> factory_function.priority
105
```

iter_factories()
Iterate through all factories identified by the factory entrypoint.

Yields *function* – A function that accepts a *Specification* and returns a *PenaltyModel*.

iter_caches()
Iterator over the PenaltyModel caches.

Yields *function* – A function that accepts a `PenaltyModel` and caches it.

Exceptions

exception FactoryException
General exception for a factory being not able to produce a penalty model.

exception ImpossiblePenaltyModel
PenaltyModel is impossible to build.

exception MissingPenaltyModel
PenaltyModel is missing from the cache or otherwise unavailable.

License

"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.

1.1.2 penaltymodel-cache

A local cache for penalty models. Serves as a factory and cache for penaltymodel.

On install, penaltymodel_cache registers an entry point that can be read by penaltymodel. By identifying itself as both a cache and a factory, it will be used automatically by any project that uses penaltymodel's `get_penalty_model` function. It will also be automatically populated

Interface

This module has the primary public-facing methods for the project.

get_penalty_model(*specification*, *database=None*)
Factory function for penaltymodel_cache.

Parameters

- **specification** (*penaltymodel.Specification*) – The specification for the desired penalty model.
- **database** (*str, optional*) – The path to the desired sqlite database file. If None, will use the default.

- **priority** (*int*) – 100

Returns Penalty model with the given specification.

Return type penaltymodel.PenaltyModel

Raises penaltymodel.MissingPenaltyModel – If the penalty model is not in the cache.

cache_penalty_model(*penalty_model*, *database=None*)
Caching function for penaltymodel_cache.

Parameters

- **penalty_model** (penaltymodel.PenaltyModel) – Penalty model to be cached.
- **database** (*str, optional*) – The path to the desired sqlite database file. If None, will use the default.

Database Utilities

Utilities for access to the sqlite cache.

cache_connect(*database=None*)
Returns a connection object to a sqlite database.

Parameters **database** (*str, optional*) – The path to the database the user wishes to connect to. If not specified, a default is chosen using *cache_file()*. If the special database name ':memory:' is given, then a temporary database is created in memory.

Returns sqlite3.Connection

insert_graph(*cur*, *nodelist*, *edgelist*, *encoded_data=None*)
Insert a graph into the cache.

A graph is stored by number of nodes, number of edges and a json-encoded list of edges.

Parameters

- **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a with statement.
- **nodelist** (*list*) – The nodes in the graph.
- **edgelist** (*list*) – The edges in the graph.
- **encoded_data** (*dict, optional*) – If a dictionary is provided, it will be populated with the serialized data. This is useful for preventing encoding the same information many times.

Notes

This function assumes that the nodes are index-labeled and range from 0 to num_nodes - 1.

In order to minimize the total size of the cache, it is a good idea to sort the nodelist and edgelist before inserting.

Examples

```
>>> nodelist = [0, 1, 2]
>>> edgelist = [(0, 1), (1, 2)]
>>> with pmc.cache_connect(':memory:') as cur:
...     pmc.insert_graph(cur, nodelist, edgelist)
```

```
>>> nodelist = [0, 1, 2]
>>> edgelist = [(0, 1), (1, 2)]
>>> encoded_data = {}
>>> with pmc.cache_connect(':memory:') as cur:
...     pmc.insert_graph(cur, nodelist, edgelist, encoded_data)
>>> encoded_data['num_nodes']
3
>>> encoded_data['num_edges']
2
>>> encoded_data['edges']
'[[0,1],[1,2]]'
```

iter_graph(*cur*)

Iterate over all graphs in the cache.

Parameters **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.

Yields *tuple* –

A 2-tuple containing:

list: The nodelist for a graph in the cache.

list: the edgelist for a graph in the cache.

Examples

```
>>> nodelist = [0, 1, 2]
>>> edgelist = [(0, 1), (1, 2)]
>>> with pmc.cache_connect(':memory:') as cur:
...     pmc.insert_graph(cur, nodelist, edgelist)
...     list(pmc.iter_graph(cur))
[([0, 1, 2], [[0, 1], [1, 2]])]
```

insert_feasible_configurations(*cur*, *feasible_configurations*, *encoded_data=None*)

Insert a group of feasible configurations into the cache.

Parameters

- **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.
- **feasible_configurations** (*dict[tuple[int]]*) – The set of feasible configurations. Each key should be a tuple of variable assignments. The values are the relative energies.
- **encoded_data** (*dict, optional*) – If a dictionary is provided, it will be populated with the serialized data. This is useful for preventing encoding the same information many times.

Examples

```
>>> feasible_configurations = {(-1, -1): 0.0, (+1, +1): 0.0}
>>> with pmc.cache_connect(':memory:') as cur:
...     pmc.insert_feasible_configurations(cur, feasible_configurations)
```

iter_feasible_configurations(*cur*)
Iterate over all of the sets of feasible configurations in the cache.

Parameters **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.

Yields *dict[tuple(int)* – number]: The feasible_configurations.

insert_ising_model(*cur*, *nodelist*, *edgelist*, *linear*, *quadratic*, *offset*, *encoded_data=None*)
Insert an Ising model into the cache.

Parameters

- **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.
- **nodelist** (*list*) – The nodes in the graph.
- **edgelist** (*list*) – The edges in the graph.
- **linear** (*dict*) – The linear bias associated with each node in nodelist.
- **quadratic** (*dict*) – The quadratic bias associated with teach edge in edgelist.
- **offset** (*float*) – The constant offset applied to the ising problem.
- **encoded_data** (*dict, optional*) – If a dictionary is provided, it will be populated with the serialized data. This is useful for preventing encoding the same information many times.

iter_ising_model(*cur*)
Iterate over all of the Ising models in the cache.

Parameters **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.

Yields *tuple* –

A 5-tuple consisting of:

list: The nodelist for a graph in the cache.

list: the edgelist for a graph in the cache.

dict: The linear biases of an Ising Model in the cache.

dict: The quadratic biases of an Ising Model in the cache.

float: The constant offset of an Ising Model in the cache.

insert_penalty_model(*cur*, *penalty_model*)
Insert a penalty model into the database.

Parameters

- **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.
- **penalty_model** (penaltymodel.PenaltyModel) – A penalty model to be stored in the database.

Examples

```
>>> import networkx as nx
>>> import penaltymodel.core as pm
>>> import dimod
>>> graph = nx.path_graph(3)
>>> decision_variables = (0, 2)
>>> feasible_configurations = {(-1, -1): 0., (+1, +1): 0.}
>>> spec = pm.Specification(graph, decision_variables, feasible_configurations,
↪dimod.SPIN)
>>> linear = {v: 0 for v in graph}
>>> quadratic = {edge: -1 for edge in graph.edges}
>>> model = dimod.BinaryQuadraticModel(linear, quadratic, 0.0, vartype=dimod.SPIN)
>>> widget = pm.PenaltyModel.from_specification(spec, model, 2., -2)
>>> with pmc.cache_connect(':memory:') as cur:
...     pmc.insert_penalty_model(cur, widget)
```

iter_penalty_model_from_specification(*cur*, *specification*)

Iterate through all penalty models in the cache matching the given specification.

Parameters

- **cur** (sqlite3.Cursor) – An sqlite3 cursor. This function is meant to be run within a `with` statement.
- **specification** (penaltymodel.Specification) – A specification for a penalty model.

Yields penaltymodel.PenaltyModel

Database Schema

```
# Copyright 2017 D-Wave Systems Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""The schema used by the sqlite database for storing the penalty models."""

schema = \
    """
    CREATE TABLE IF NOT EXISTS graph(
        num_nodes INTEGER NOT NULL,  -- for integer-labeled graphs, num_nodes encodes
↪all of the nodes
        num_edges INTEGER NOT NULL,  -- redundant, allows for faster selects
        edges TEXT NOT NULL,  -- json list of lists, should be sorted (with each edge
↪sorted)
        id INTEGER PRIMARY KEY,
        CONSTRAINT graph UNIQUE (
            num_nodes,
```

(continues on next page)

(continued from previous page)

```
            edges));

    CREATE TABLE IF NOT EXISTS feasible_configurations(
        num_variables INTEGER NOT NULL,
        num_feasible_configurations INTEGER NOT NULL,
        feasible_configurations TEXT NOT NULL,
        energies TEXT NOT NULL,
        id INTEGER PRIMARY KEY,
        CONSTRAINT feasible_configurations UNIQUE (
            num_variables,
            num_feasible_configurations,
            feasible_configurations,
            energies));

    CREATE TABLE IF NOT EXISTS ising_model(
        linear_biases TEXT NOT NULL,
        quadratic_biases TEXT NOT NULL,
        offset REAL NOT NULL,
        max_quadratic_bias REAL NOT NULL,
        min_quadratic_bias REAL NOT NULL,
        max_linear_bias REAL NOT NULL,
        min_linear_bias REAL NOT NULL,
        graph_id INTEGER NOT NULL,
        id INTEGER PRIMARY KEY,
        CONSTRAINT ising_model UNIQUE (
            linear_biases,
            quadratic_biases,
            offset,
            graph_id),
        FOREIGN KEY (graph_id) REFERENCES graph(id) ON DELETE CASCADE);

    CREATE TABLE IF NOT EXISTS penalty_model(
        decision_variables TEXT NOT NULL,
        classical_gap REAL NOT NULL,
        ground_energy REAL NOT NULL,
        feasible_configurations_id INT,
        ising_model_id INT,
        id INTEGER PRIMARY KEY,
        FOREIGN KEY (feasible_configurations_id) REFERENCES feasible_
↪configurations(id) ON DELETE CASCADE,
        FOREIGN KEY (ising_model_id) REFERENCES ising_model(id) ON DELETE CASCADE,
        CONSTRAINT ising_model UNIQUE (
            decision_variables,
            feasible_configurations_id,
            ising_model_id));

    CREATE VIEW IF NOT EXISTS penalty_model_view AS
    SELECT
        num_variables,
        num_feasible_configurations,
        feasible_configurations,
        energies,

        num_nodes,
        num_edges,
        edges,
```

(continues on next page)

(continued from previous page)

```
        linear_biases,
        quadratic_biases,
        offset,
        max_quadratic_bias,
        min_quadratic_bias,
        max_linear_bias,
        min_linear_bias,

        decision_variables,
        classical_gap,
        ground_energy,
        penalty_model.id
    FROM
        ising_model,
        feasible_configurations,
        graph,
        penalty_model
    WHERE
        penalty_model.ising_model_id = ising_model.id
        AND feasible_configurations.id = penalty_model.feasible_configurations_id
        AND graph.id = ising_model.graph_id;
    """
```

Cache Information

cache_file(*app_name='dwave-penaltymodel-cache'*, *app_author='dwave-systems'*, *filename='penaltymodel_cache_v0.4.0.db'*)

Returns the filename (including path) for the data cache.

The path will depend on the operating system, certain environmental variables and whether it is being run inside a virtual environment. See homebase.

Parameters

- **app_name** (*str, optional*) – The application name. Default is given by *APPNAME*.
- **app_author** (*str, optional*) – The application author. Default is given by *APPAUTHOR*.
- **filename** (*str, optional*) – The name of the database file. Default is given by *DATABASENAME*.

Returns The full path to the file that can be used as a cache.

Return type str

Notes

Creates the directory if it does not already exist.

If run inside of a virtual environment, the cache will be stored in */path/to/virtualenv/data/app_name*

APPNAME = 'dwave-penaltymodel-cache'

The application name is used to determine the cache location.

APPAUTHOR = 'dwave-systems'

The application author is used to determine the cache location.

DATABASENAME = 'penaltymodel_cache_v0.4.0.db'
The name for the sqlite database itself. Based on the version of the package.

License

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.

1.1.3 penaltymodel-maxgap

Generates penalty models using smt solvers. Serves as a factory and cache for penaltymodel.

On install, penaltymodel_maxgap registers an entry point that can be read by penaltymodel. It will be used automatically by any project that uses penaltymodel's `get_penalty_model` function.

Interface

get_penalty_model(*specification*)

Factory function for penaltymodel_maxgap.

Parameters

- **specification** (*penaltymodel.Specification*) – The specification for the desired penalty model.
- **priority** (*int*) – -100

Returns Penalty model with the given specification.

Return type `penaltymodel.PenaltyModel`

Raises `penaltymodel.ImpossiblePenaltyModel` – If the penalty cannot be built.

SMT Generation

generate(*graph*, *feasible_configurations*, *decision_variables*, *linear_energy_ranges*, *quadratic_energy_ranges*, *min_classical_gap*, *smt_solver_name=None*)

Generates the Ising model that induces the given feasible configurations. The code is based on the papers[1] and[2].

Parameters

[1] Bian et al., "Discrete optimization using quantum annealing on sparse Ising models", https://www.frontiersin.org/articles/10.3389/fphy.2014.00056/full

[2] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy "Mapping constrained optimization problems to quantum annealing with application to fault diagnosis" https://arxiv.org/pdf/1603.03111.pdf

- **graph** (*nx.Graph*) – The target graph on which the Ising model is to be built.
- **feasible_configurations** (*dict*) – The set of feasible configurations of the decision variables. The key is a feasible configuration as a tuple of spins, the values are the associated energy.
- **decision_variables** (*list/tuple*) – Which variables in the graph are assigned as decision variables.
- **linear_energy_ranges** (*dict, optional*) – A dict of the form {v: (min, max), ...} where min and max are the range of values allowed to v.
- **quadratic_energy_ranges** (*dict*) – A dict of the form {(u, v): (min, max), ...} where min and max are the range of values allowed to (u, v).
- **min_classical_gap** (*float*) – The minimum energy gap between the highest feasible state and the lowest infeasible state.
- **smt_solver_name** (*str/None*) – The name of the smt solver. Must be a solver available to pysmt. If None, uses the pysmt default.

Returns

A 4-tuple containing:

dict: The linear biases of the Ising problem.

dict: The quadratic biases of the Ising problem.

dimod.BinaryQuadraticModel

float: The classical energy gap between ground and the first excited state.

Return type tuple

Raises ImpossiblePenaltyModel – If the penalty model cannot be built. Normally due to a non-zero infeasible gap.

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.

1.1.4 penaltymodel-mip

Generates penalty models using Google Optimization Tools' Mixed-Integer Programming capability. Serves as a factory and cache for penaltymodel.

On install, penaltymodel-mip registers an entry point that can be read by penaltymodel. It will be used automatically by any project that uses penaltymodel's `get_penalty_model` function.

Interface

get_penalty_model(*specification*)
Factory function for penaltymodel-mip.

Parameters

- **specification** (*penaltymodel.Specification*) – The specification for the desired penalty model.
- **priority** (*int*) – -100

Returns Penalty model with the given specification.

Return type penaltymodel.PenaltyModel

Raises penaltymodel.ImpossiblePenaltyModel – If the penalty cannot be built.

Mixed Integer (Linear) Programming Generation

generate_bqm(*graph*, *table*, *decision*, *linear_energy_ranges=None*, *quadratic_energy_ranges=None*, *min_classical_gap=2*, *precision=7*, *max_decision=8*, *max_variables=10*, *return_auxiliary=False*)
Get a binary quadratic model with specific ground states.

Parameters

- **graph** (Graph) – Defines the structure of the generated binary quadratic model.
- **table** (*iterable*) – Iterable of valid configurations (of spin-values). Each configuration is a tuple of variable assignments ordered by *decision*.
- **decision** (*list/tuple*) – The variables in the binary quadratic model which have specified configurations.
- **linear_energy_ranges** (*dict, optional*) – Dict of the form {v: (min, max, ...} where min and max are the range of values allowed to v. The default range is [-2, 2].
- **quadratic_energy_ranges** (*dict, optional*) – Dict of the form {(u, v): (min, max), ...} where min and max are the range of values allowed to (u, v). The default range is [-1, 1].
- **min_classical_gap** (*float*) – The minimum energy gap between the highest feasible state and the lowest infeasible state.
- **precision** (*int, optional, default=7*) – Values returned by the optimization solver are rounded to *precision* digits of precision.
- **max_decision** (*int, optional, default=4*) – Maximum number of decision variables allowed. The algorithm is valid for arbitrary sizes of problem but can be extremely slow.
- **max_variables** (*int, optional, default=4*) – Maximum number of variables allowed. The algorithm is valid for arbitrary sizes of problem but can be extremely slow.
- **return_auxiliary** (*bool, optional, False*) – If True, the auxiliary configurations are returned for each configuration in table.

Returns

`dimod.BinaryQuadraticModel`: The binary quadratic model.

float: The classical gap.

If return_auxiliary is True:

`dimod.BinaryQuadraticModel`: The binary quadratic model.

float: The classical gap.

dict: The auxiliary configurations, keyed on the configurations in table.

Return type If return_auxiliary is False

Raises `ImpossiblePenaltyModel` – If the penalty model cannot be built. Normally due to a non-zero infeasible gap.

License

"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.

1.1.5 penaltymodel-lp

Generates penalty models using scipy.optimize's Linear Programming capability. Serves as a factory and cache for penaltymodel.

On install, penaltymodel-lp registers an entry point that can be read by penaltymodel. It will be used automatically by any project that uses penaltymodel's `get_penalty_model` function.

Interface

get_penalty_model(*specification*)
Factory function for penaltymodel-lp.

Parameters

- **specification** (*penaltymodel.Specification*) – The specification for the desired penalty model.
- **priority** (*int*) – -100

Returns Penalty model with the given specification.

Return type penaltymodel.PenaltyModel

Raises `penaltymodel.ImpossiblePenaltyModel` – If the penalty cannot be built.

Linear Programming Generation

generate_bqm(*graph*, *table*, *decision_variables*, *linear_energy_ranges=None*, *quadratic_energy_ranges=None*, *min_classical_gap=2*, *catch_warnings=True*)

Parameters

- **graph** – A networkx.Graph
- **table** – An iterable of valid spin configurations. Each configuration is a tuple of variable assignments ordered by *decision*.
- **decision_variables** – An ordered iterable of the variables in the binary quadratic model.
- **linear_energy_ranges** – Dictionary of the form {v: (min, max), ...} where min and max are the range of values allowed to v. The default range is [-2, 2].
- **quadratic_energy_ranges** – Dict of the form {(u, v): (min, max), ...} where min and max are the range of values allowed to (u, v). The default range is [-1, 1].
- **min_classical_gap** – A float. The minimum energy gap between the highest feasible state and the lowest infeasible state.

get_item(*dictionary*, *tuple_key*, *default_value*)

Grab values from a dictionary using an unordered tuple as a key.

Dictionary should not contain None, 0, or False as dictionary values.

Parameters

- **dictionary** – Dictionary that uses two-element tuple as keys
- **tuple_key** – Unordered tuple of two elements
- **default_value** – Value that is returned when the tuple_key is not found in the dictionary

License

"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.

1.2 Installation

1.2.1 Install penaltymodel-core

To install:

```
pip install penaltymodel
```

To build from souce:

```
cd penatlymodel_core
pip install -r requirements.txt
python setup.py install
```

1.2.2 Install penaltymodel-cache

To install:

```
pip install penaltymodel_cache
```

To build from source:

```
cd penaltymodel_cache
pip install -r requirements.txt
pip install -e ../penaltymodel_core/
python setup.py install
```

1.2.3 Install penaltymodel-maxgap

To install:

```
pip install penaltymodel_maxgap
```

To build from source:

```
cd penaltymodel_maxgap
pip install -r requirements.txt
pip install -e ../penaltymodel_core/
python setup.py install
```

Note that this library will not function without smt solvers installed. The solvers are accessed through the pysmt package.

In the standard setup (`pip install` or `setup.py install` above), Z3 solver is installed auto-magically. See the accompanying pysmt documentation for installing other smt solvers.

In development mode (`pip install -e` or `setup.py develop`) solvers are not installed. Check pysmt documentation to see how to do it manually.

1.2.4 Install penaltymodel-mip

To install:

```
pip install penaltymodel-mip
```

To build from souce:

```
cd penaltymodel_mip
pip install -r requirements.txt
python setup.py install
```

1.2.5 Install penaltymodel-lp

To install:

```
pip install penaltymodel-lp
```

To build from souce:

```
cd penaltymodel_lp
pip install -r requirements.txt
python setup.py install
```

1.3 License

"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.

Python Module Index

c

l

m

Index

M

P

R

S

V