Quanfluence Ising Machine Quick Start

Version 0.3.0: 12-Feb-2025

1. Contents of the distribution package

This distribution package contains the following files:

1. README_quickstart (this file)
2. *bqm_example.py*
3. *upload_example.py*
4. *run_example.py*

This is a quick start guide to enable access to the Quanfluence Ising Server.

The *quanfluence-sdk* is the python library for calling the Quanfluence Ising Machine in the cloud. The `QuanfluenceClient` class allows you to create, configure and run an Ising device on a remote server. This documentation provides detailed instructions on how to use the class, including prerequisites, function parameters, return values, and usage instructions. This quick start package contains *example.py* file which is preconfigured with user credentials, device ID and some initial configuration for the Ising Machine. Users can make modifications to the *bqm_example.py* file and start running their Ising problem or upload their QUBO file once using *upload_example.py* and run it multiple times using *run_example.py* as well as make suitable modifications to the machine configuration to optimize performance.

2. Prerequisites

Ensure you have the following libraries installed:

- requests
- quanfluence-sdk

You can install the required libraries using pip:
pip install quanfluence-sdk requests

The `QuanfluenceClient` class requires the librar(ies) listed in Prerequisites

You are provided with the preconfigured credentials and device id to enable your developer access.

3. Running your QUBO Using *bqm_example.py*

Copy *bqm_example.py* file to your working directory. Update the section marked "USER SECTION" for including your QUBO. Run the *bqm_example.py* file.

The configuration parameter such as alpha, beta, beta_decay, noise for the ISING machine can be read back and written into the DEVICE. The appropriate sections are marked out in the *bqm_example.py* file.

4 a). Uploading and running a QUBO file Using *upload_example.py* and *run_example.py* (Only required if your QUBO is a file)

In the *upload_example.py* file just enter the path to your QUBO file in the *upload_qubo()* method and execute the file. This needs to be done only once. Remember the filename obtained after uploading the QUBO file.

Now, enter the filename obtained after uploading in *execute_qubo_file()* method of the *run_example.py* file and execute the file to run the QUBO on the server.

4. Setting alpha, beta and configuration parameters

The optimum solution of the ISING machine requires setting configuration parameters like alpha, beta, noise. Please refer to time-multiplexed coherent Ising Machine [1] or consult with the Quanfluence team.

5. Limitations

The current supported BQM on the Ising machine has the following limitations

- The nodes in the BQM (QUBO or Ising models) should always be indexed from 0 to N-1 where N is the number of nodes
- Disconnected nodes with 0 self-weight are not allowed
- The values in QUBO passed in dictionary format should always be 32 - bit floating point numbers.
  The QUBO from BQM given by dimod by default has Float 64 values that need to be converted to Float before execution on the Ising machine. An example of converting QUBO values from Float 64 to Float is shown file – *bqm_example.py*.

6. Class Methods

Refer the documentation : quanfluence-sdk · PyPI

1. *signin(self, username, password)*

Authenticates a user and retrieves an access token. This is already preconfigured.

Parameters:

username
The username of the user.
password
The password of the user.

Returns:
out
The API response with the access token.

2. *get_device(self, device_id)*

Retrieves details of a specific device.

Parameters:
device_id
The ID of the device to retrieve.

Returns:
out
The API response.

3. *update_device(self, device_id, updates)*

Updates details of an existing device.

Parameters
device_id
The ID of the device to update.
Parameters:
title
The title of the device.
description
The description of the device.
type
The type of the device (local or remote).
public_ipv4_address
The public IPv4 address of the device.
private_ipv4_address
The private IPv4 address of the device.
iters
The number of iterations for the device.
runs
The number of runs for the device.

alpha
The alpha parameter for the device.
beta
The beta parameter for the device.
beta_decay
The beta decay parameter for the device.
noise_stdev
The noise standard deviation for the device.
runtime
The runtime for the device.

Returns:
out
The API response.

Returns:
out
The API response.

4. *upload_device_qubo(self, device_id, filename)*

Uploads a QUBO file to a specific device.

Parameters:
device_id
The ID of the device to upload the QUBO.
filename
The filename of the QUBO file.

Returns:
out
The API response.

5. *execute_device_qubo_input(self, device_id, qubo)*

Executes a QUBO on a specific device.

Parameters:
device_id
The ID of the device to execute the QUBO.
qubo
The QUBO to execute.

Returns:
out
The API response.

6. *execute_device_qubo_file(self, device_id, filename)*

Executes a QUBO on a specific device.

Parameters:

device_id
The ID of the device to execute the QUBO.

qubo
The QUBO to execute.

Returns
The API response.

Note:

The Quanfluence Client does not require Dimod package for execution. If your QUBO is in dictionary format, you can pass it directly to the Quanfluence Server.

Dimod package is required only if your QUBO is in BQM format.

You can refer to the dimod documentation to get a better understanding of dimod and BQM here [2].

Reference:

[1] A. Prabhakar, P. Shah, U. Gautham, V. Natarajan, V. Ramesh,
N. Chandrachoodan and S. Tayur, Optimization with photonic wave based annealers, rspa.royalsocietypublishing.org

[2] dimod — Ocean Documentation 8.0.1 documentation (dwavesys.com)

[3] quanfluence-sdk · PyPI