User's Manual for SQBM+™ V2

Note

Table of Contents

1. Introduction

SQBM+ (Simulated Quantum-inspired Bifurcation Machine plus) is a set of solvers enabling users to quickly obtain nearly optimal solutions for large combinatorial optimization problems. SQBM+'s solver allows users to obtain optimal or good solutions. SQBM+ is developed based on the theory described in the following papers:

- Goto, H., Tatsumura, K., & Dixon, A. R. (2019) Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, *Science Advances*, 5(4), DOI:10.1126/sciadv.aav2372
- Goto, H., Endo, K., Suzuki, M., Sakai, Y., Kanao T., Hamakawa Y., Hidaka, R., Yamasaki, M., & Tatsumura, K. (2021) High-performance combinatorial optimization based on classical mechanics, *Science Advances*, 7(6), DOI:10.1126/sciadv.abe7953

SQBM+ is provided as a virtual machine image for use on a virtual machine instance (SQBM+ server instance) or a RPM package. SQBM+ is configured as shown in Figure 1.

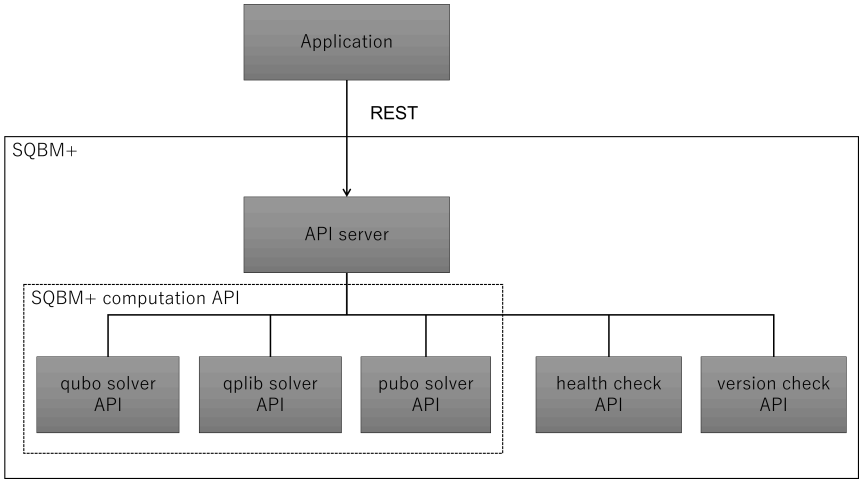

Figure 1. Functional block diagram of SQBM+

SQBM+ has the following main characteristics:

- Offers SQBM+ computation API using RESTful API.
- Provided with solvers and interfaces compatible with various optimization problems and data formats. Currently, the following solvers are available. For details, see the respective sections of each solver.
  - qubo solver API: a solver corresponding to the qubo model
  - qplib solver API: a solver that solves a linear constrained quadratic programming problem.
  - pubo solver API: a solver that solves a problem with higher order terms up to order 4.
- Quickly obtains optimal or nearly optimal solutions for large optimization problems[1].
- Easy to use. Complicated parameter setting is not required.

This document is organized as follows:

- Overview of SQBM+
- SQBM+ computation API
- Health check API
- Version check API
- System configuration parameters

First, the Chapter "Overview of SQBM+" presents an overview of APIs provided by SQBM+, where users can gain an overview of how to use APIs provided by SQBM+ and their responses. This Chapter is followed by descriptions of a total of three APIs, consisting of the SQBM+ computation API, the health check API, and the version check API. By reading chapters describing each of the three APIs after reading the overview, users can learn how to use them in detail.
The Chapter System configuration parameters describes system settings for SQBM+. Users can use SQBM+ without any system settings. However, in some cases, default settings need to be modified to handle much larger problems or to modify the computation time limit. If so, read this Chapter and determine whether the default settings need modification.

2. Overview of SQBM+

SQBM+ provides the following three APIs:

- SQBM+ computation API: API which finds a solution to a combinatorial optimization problem.
- Health check API: API for checking the operation of the SQBM+ computation API.
- Version check API: API for checking the SQBM+ version you are using.

Each of these three APIs can be used by sending an HTTP request to the SQBM+ server from the client.
A response from each API is returned as an HTTP response.
Figure 2 is an example of using the SQBM+ computation API where the client using the API sends an HTTP request to the SQBM+ server by running the curl command. In this example, the client sends an HTTP request specifying that the qubo solver API be used to solve an optimization problem. The SQBM+ server receiving this HTTP request performs computation and returns a solution found to the client. The response is in JSON format.

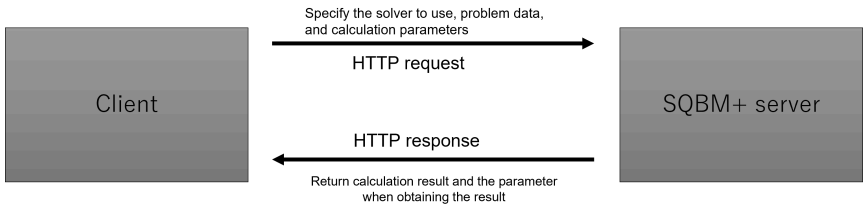

Figure 2. Example of using the SQBM+ computation API

Table 1 summarizes APIs provided by SQBM+. For details of each API, refer to Sections SQBM+ computation API, Health check API, and Version check API.

Table 1. SQBM+ API information

API type	HTTP method	URL path	Port	Character code
SQBM+ computation API	POST	http://{IP address of the server}:{port number}/solver/{solver} {solver}: Select one of qubo, qplib, pubo as a solver to use. {port number}: The default is 8000.	8000 (default)	UTF-8
health check API	GET	http://{IP address of the server}:{port number}/healthcheck {port number}: The default is 8000.	8000 (default)	UTF-8
version check API	GET	http://{IP address of the server}:{port number}/version {port number}: The default is 8000.	8000 (default)	UTF-8

Each SQBM+ API has the following characteristics:

- SQBM+ has no session information.
- SQBM+ does not retain data included in a request as well as response data sent to the user. Note that you cannot ask for the result of past requests and that no data is retained.
- Each SQBM+ API and function can be used by sending a corresponding HTTP request. See the URL path in Table 1 to access each API and function.

Before using the SQBM+ computation API, select the solver that suits the problem you want to solve.
In SQBM+ computation API, you can set parameters related to computation control as a query string in an HTTP request.
Set problem data to be solved by the SQBM+ computation API as an HTTP request body.
SQBM+ computation API defines specifications common to all solvers as in Figure 3, in addition to solver-specific specifications. Specifications regarding responses and available parameters are categorized into two; those that are common to all solvers and those specific to each solver. Also, the format of problem data that can be set as a request body is determined by each solver. The user needs to read Specifications common to all solvers and understand the specifications of each solver.

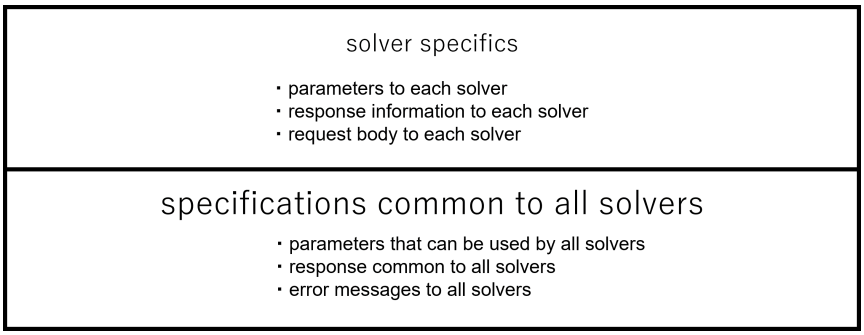

Figure 3. Specification hierarchy of the SQBM+ computation API

3. SQBM+ computation API

As one of its main APIs, SQBM+ provides a solver to solve a combinatorial optimization problem. User need to check each section of each solver which suits the problem you want to solve. Each solver accepts problem data in various formats. The user does not need to specify the data format in an HTTP request, because each solver can automatically identify the format of problem data. For input formats of each solver, each solver section will be helpful.

The following information is required for SQBM+ to solve a problem:

- solver to be used and problem data (required)
- parameter specification that controls how SQBM+ solves a problem (optional)

The user needs to review the Section Specifications common to all solvers to understand the specifications of HTTP requests and responses common to all solvers first; then, he/she can better understand the specifications and response details specific to each solver by reviewing the respective sections on specifications of each solver.
If there is no parameter specification that controls how SQBM+ solves a problem, then either the setting value of system configuration parameters is applied, or problem-solving is automatically adjusted within SQBM+. For these reasons, parameter specification is not required. The SQBM+ computation API provides a list of parameters used to solve a problem in the response. This allows users to reference parameter information.
Parameters related to computational sequence control play a major role in controlling the SQBM+ computation API. These parameters determine an outline of how the SQBM+ computation API finds a solution. The Section Computation structure of the SQBM+ computation API describes these parameters.

3.1. Computation structure of the SQBM+ computation API

SQBM+ computation API is capable of controlling computational sequences in two different ways; one is to control the number of solutions to a problem that is solved, the other is to control the number of times the SB algorithm to be performed is processed.

- The SQBM+ computation API is capable of specifying the number of solutions to a problem that is iteratively solved.
  It is specified through the 'loops' parameter to be described later. The number of solutions the SQBM+ computation API finds is determined by the number of GPU devices in your environment and the number of multi-processors within GPUs. The API iteratively searches for optimal solutions until at least as many solutions as 'loops' × 'the number of GPU devices' × 'the number of multi-processors' are obtained.
  The default behavior is that the SQBM+ computation API returns the best of these solutions to the user, but it is also possible to return the specified number of solutions in descending order of evaluation values. For details, see the respective sections on System configuration parameters and Request parameters.
- The SQBM+ computation API is also capable of specifying the number of times the SB algorithm to be iteratively performed is processed.
  It is specified through the 'steps' parameter to be described later. The greater the number of 'steps', the longer the time spent on searching for one optimal solution.

The number of 'steps' and 'loops' affects the accuracy of solutions to search for and computation time. The SQBM+ computation API has an auto setup functionality that helps you find better parameter settings. For details, see Request parameters.
The total number of solutions that the SQBM+ computation API has found is expressed as the parameter 'runs'. Each solution is counted as one run, which is returned as a response from the SQBM+ computation API. For details, see Response specifications. Figure 4 shows the computation structure of the SQBM+ computation API.

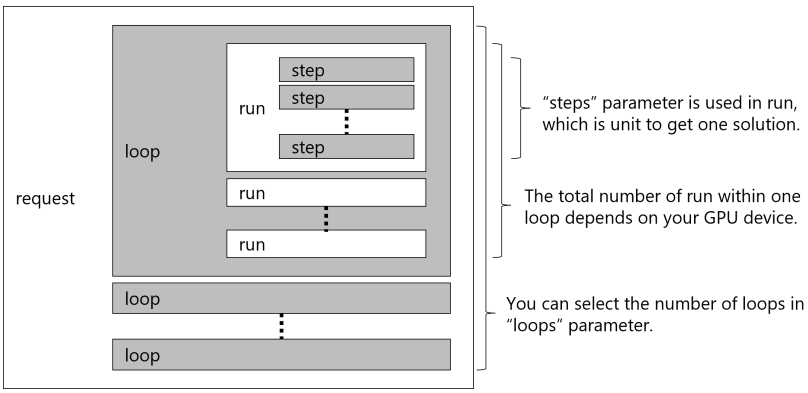

Figure 4. Computation structure of the SQBM+ computation API

3.2. Specifications common to all solvers

A request to the SQBM+ computation API is made using a RESTful API.
The HTTP method used as an RESTful API, URL configuration, and parameters common to all solvers are described in Request specifications right below.

3.2.1. Request specifications

3.2.1.1. HTTP request

- HTTP method to use: POST
- URL configuration: http://{ip}:{port}/solver/{solver}?{parameter}
  - ip: IP address of the machine on which the SQBM+ computation API is operating.
  - port: port number of the SQBM+ server
  - solver: qubo, qplib or pubo
  - parameter: query string consisting of key-value pairs, where an equal sign (=) is used to connect each key and its value. For available parameter types and details, see Request parameters and specifications specific to each solver. Default values are used if parameter values are not set.

3.2.1.2. Request parameters

Table 2. Parameters common to all solvers

Name	Default	Description
steps	0	Specify 'steps' described in Computation structure of the SQBM+ computation API. The value 0 (zero) means auto step specification, where the SQBM+ computation API performs computation by automatically changing the number of 'steps' each time the API finds a solution. Specify an integer between 0 and 100,000,000.
loops	0	Specify 'loops' described in Computation structure of the SQBM+ computation API. If 0 (zero) is specified, 'loops' is set to 100,000,000. Specify an integer between 0 and 100,000,000.
timeout	10 seconds The default value can be modified using the system configuration file.	Specify the maximum computation time (timeout) in seconds. For the Marketplace edition, this value is the upper limit of the computation time that is charged as the fee of SQBM+. When the computation time reaches the upper limit before completing the computation specified by 'loops', the computation ends at that point. In this case, the execution result will be the best solution obtained thus far. Apart from the limit set by the computation parameter 'timeout', the system sets an upper limit for computation time. For details, refer to the parameter timeout_upper_bound in System configuration parameters. If 'timeout' is equal to or greater than timeout_upper_bound, the value of timeout_upper_bound is used as 'timeout'. If the set time is long, the maximum number of solutions that SQBM+ can obtain may be reached and the calculation may be completed within the timeout time. Again, the best solution obtained so far is the execution result. Specify a value greater than 0 (zero).
maxwait	60 seconds The default value can be modified using the system configuration file.	Specify the maximum waiting time in seconds for each computation request. A computation request accepted by the API server waits in a queue for the preceding requests being processed to be completed. If the specified maximum waiting time elapses during this wait, the computation request ends with a timeout error. Specify a value greater than 0 and less than or equal to 100,000.
target	unspecified	If the parameter 'target' is specified, computation will stop when the evaluation value of an objective function reaches 'target'. If 0 (zero) is specified for the parameter 'loops', solving will be repeated either until the objective function reaches the value specified in the parameter 'target' or until a timeout occurs.
maxout	1	Specify the upper limit of the number of solutions to be outputted. Until the limit specified by 'maxout' is reached, the SQBM+ computation API outputs the obtained solutions in descending order of the value of the objective function. Specify an integer equal to or greater than one.

3.2.1.3. Problem data specification

Create combinatorial optimization problem data in a data format supported by each solver.
However, the specified data is rounded to a decimal within the range represented by 32bit or 64bit.
Combinatorial optimization problem data can be set in the following two ways.

- Setting in request body
- Specify problem file path in URI

3.2.1.3.1. Settings in the request body

Set the problem data if you pass the problem data in the request body.

3.2.1.3.2. Problem file path specification in URI

You can place the combinatorial optimization problem data file in a location accessible by the SQBM+ server, set the file path as an HTTP header, and read the problem data into SQBM+.
Please set the header information of Table 3 as the header.
Set X-Input-Data-URI when specifying the file path.
Set Content-Location when downloading from an HTTPS server (S3, etc.).

Table 3. Request headers to be set

Header field	Description
X-Input-Data-URI	absolute file path
Content-Location	URL of the file located on the HTTPS server

3.2.2. Response specifications

3.2.2.1. Response status common to all solvers

If computation completes successfully, an HTTP status code 200 is returned.
If it abends, an HTTP status code other than 200 is returned.
For details, see Table 6.

3.2.2.2. Response header common to all solvers

When computation completes successfully, the header in Table 4 is specified.

Table 4. Response header fields and values to be set

Header field	Description
Content-Type	application/json
X-Calculation-Time	Actual computation time. Same as 'time' in the response body.
X-ID	Internally assigned request ID.

When there is a calculation error, "text/html; charset=utf-8" is set as the Content-Type value, and there is no X-Calculation-Time header or X-ID header.

3.2.2.3. Response body common to all solvers

Table 5 gives a list of properties that are stored in the response body when computation completes successfully. The results are returned in JSON format. The properties in Table 5 are the properties that all solver have in common.

Table 5. Properties of computation results common to all solvers

Property name	Description
id	Internally assigned request ID.
time	Actual computation time in seconds, not including overheads, such as data transfer time (in seconds). For the Marketplace edition, this value is the quantity of "computation time" that is charged as the fee of SQBM+. However, the charged quantity is up to the value of `timeout` request parameter.
wait	Processing latency from the time when the SQBM+ server loads a problem until the SQBM+ computation is started (in seconds).
message	Messages about completion conditions of the SQBM+ computation API. One of the five message strings (finished, timeout, reached, max results, and saturated) is set. Their specifications are as follows: - "finished": Completed the computation of the specified number of steps and loops. - "timeout": Stopped the computation because the computation time has reached its timeout. - "reached": Stopped the computation because the evaluation value has reached the value specified in the parameter 'target'. - "max results": Obtained the maximum number of solutions required in one request. - "saturated": With no hope of getting a better solution, the calculation converged.
runs	Number of solutions found.
value	Evaluation value computed on the basis of solutions represented by the 'result' in this table.
result	Best solution found.
count	the number of times the best solution is obtained.
param	values of parameter settings used when the best solution is obtained. JSON format. Parameters displayed vary depending on which solver and algorithm to use.
others	A list consisting of a set of each evaluation value for other solutions, a vector of variables, 'param', and the number of times that particular solution is obtained. The total number of solutions included here is equal to or less than 'request parameters maxout minus 1'. Refer to Example of outputting multiple solutions to see examples.

Note that the input data to the SQBM+ computation API and the output data from the SQBM+ computation API are not stored in SQBM+.

3.2.2.4. Error messages

If an anomaly occurs in the SQBM+ computation API, an HTTP response status code other than 200 is returned. An HTTP response body contains a message showing the cause of an error. Table 6 lists error messages.

Table 6. List of error messages

Status code	Message	Description
400	error message regarding input parameter, input problem data	The input parameter, input problem data is incorrect. The message contains detailed information.
400	no result	No feasible solution found. It is possible that 'timeout' is too short, or linear constraint conditions are too strict.
403	The evaluation period has passed.	The trial period has expired. Occurs only in evaluation version.
403	solver is unavailable because it has not been completed usage metering	This message will be returned when SQBM+ have not finished metering for usage. Occurs only in the Marketplace edition. It may take a few minutes to finish it. Please try again in a few minutes.
403	solver is unavailable due to a failed connection to the Marketplace	This message will be returned when SQBM+ fail to connect to Marketplace. Occurs only in the Marketplace edition. Please check your network configuration.
403	solver is unavailable due to the access denied to the Marketplace	This message will be returned when SQBM+ don't have authority to access Marketplace. Occurs only in the Marketplace edition. Please check IAM role settings.
403	solver is unavailable because usage metering failed for more than 2 hours	This message will be returned when SQBM+ have failed to meter usage for more than 2 hours. Occurs only in the Marketplace edition. Please check your network configuration and IAM role. If you can't resolve problem, then reboot the SQBM+ server instance.
404	message about URL anomaly	The requested URL is invalid.
413	payload too large	The size of a problem file has exceeded payload_limit. For details, see System configuration parameters.
413	content too large	The size of a problem file has exceeded payload_limit. For details, see System configuration parameters.
500	error message about the server's internal processing. The specific messages vary case by case	The occurrence of this error means that an error unexpected by the SQBM+ server has occurred. Contact the server administrator.
503	busy	The number of requests that can be accepted at the same time.
503	engine stopped	The engine has stopped. In this case, SQBM+ automatically attempts a restart. For details about this error, see sbm-engine.log in the SQBM+ server instance. For details, see Log files.
504	timeout	The wait time specified by maxwait has elapsed, SQBM+ computation for a request has ended.

3.3. qubo solver API

The qubo solver API solves a combinatorial optimization problem represented by the qubo model.

3.3.1. Request specifications of the qubo solver API

3.3.1.1. HTTP request

In http://{ip}:{port}/solver/{solver}?{parameter}, select qubo for {solver}.

3.3.1.2. Request header

Table 7. Request header field and a value that need to be set

Header field	Description
Content-Type	application/octet-stream

3.3.1.3. Request parameters

In addition to the parameters common to all solvers (Table 2), solver-specific parameters listed in Table 8 can be used in the qubo solver API.

Table 8. Solver-specific parameters

Parameter	Default	Description
dt	0	Specify the time step size used in the time evolution of the SB algorithm. 0 (zero) denotes the auto-adjustment specification of the time step size, in which the SQBM+ computation API performs computation by automatically and internally changing dt. Specify a value between 0.0 and 1.5.
C	0	Corresponds to ξ_0 used in a paper by Goto, Tatsumura, & Dixon (2019), which proposes a theory on which SQBM+ is based. If 0 (zero) is specified, the value of C will be auto-adjusted. Specify a single-precision floating point number equal to or greater than 0 (zero).
algo	0	Specify an algorithm that is implemented within SQBM+ including the SQBM+ computation algorithms proposed in a paper by Goto et al. (2021), such as 15 (bSB) and 20 (dSB). And you can also use algorithm 25 and 30. For algorithms that can be specified, see Table 9. Computation results may vary depending on which algorithm is used. If 0 (zero) is specified, SQBM+ searches for a solution using various algorithms.
algos	-	Multiple computation algorithms can be specified by using regular expressions. If this parameter is specified, multiple algorithms specified by 'algos' will be used instead of by 'algo'. Algorithms can be specified as shown in the example below. Note that not all regular expressions can be used. Specifically, `http://localhost:8000/solver/qplib?timeout=1&algos=1.` needs to be encoded as `http://localhost:8000/solver/qplib?timeout=1&algos=1%2e` .
blocks	0	Specify the number of blocks of GPUs used to find a solution. If 0 (zero) is specified, the value of 'blocks' will be auto-adjusted. Specify an integer between 0 and 40.
multishot	0	When you specify multishot, SQBM+ will get multiple solutions, which start from different initail decision variable values but other parameters are the same. The number of solution is multishot number. Default value is 0, which automatically decide multishot from problem size. If multishot > 1, SQBM+ will have less overhead. Specify an integer between 0 and 10.

Table 9. Available algorithms

Algorithm	Description
15	bSB, the SQBM+ computation algorithm proposed in a paper by Goto et al. (2021).
151	In addition to the processing of algo=15, C is automatically adjusted for each SB algorithm updates step. This may improve accuracy over specifying a fixed C.
154	In addition to processing of algo=15, the impact size of the QUBO coefficient matrix on the SB algorithm is autoscaled. Precision may improve when the distribution of eigenvalues of the QUBO coefficient matrix is not even.
155	Both 151 and 154 are performed.
20	dSB, the SQBM+ computation algorithm proposed in a paper by Goto et al. (2021).
201	In addition to the processing of algo=20, C is automatically adjusted for each SB algorithm updates step. This may improve accuracy over specifying a fixed C.
204	In addition to the processing of algo=20, the impact size of the QUBO coefficient matrix on the SB algorithm is autoscaled. Precision may improve when the distribution of eigenvalues of the QUBO coefficient matrix is not even.
205	Both 201 and 204 are performed.
25	Extension of 15. This algorithm more likely escape from local minimum.
251	Extension of 151. This algorithm more likely escape from local minimum.
254	Extension of 154. This algorithm more likely escape from local minimum.
255	Extension of 155. This algorithm more likely escape from local minimum.
30	Extension of 20. This algorithm more likely escape from local minimum.
301	Extension of 201. This algorithm more likely escape from local minimum.
304	Extension of 204. This algorithm more likely escape from local minimum.
305	Extension of 205. This algorithm more likely escape from local minimum.

3.3.1.4. Problem data specification

In the problem data specification, a coefficient matrix corresponding to problem data needs to be represented in the QUBO format. The energy in a given problem is calculated by using this QUBO matrix. The energy to be minimized is represented by the following formula:

$$E = \sum_{i \geq j} Q_{ij} \cdot x_i \cdot x_j$$

where Q_{ij} denotes the element of the matrix, and $x_i \in \{0, 1\}$ denotes a qubo variable. The formula above can be converted to the formula below:

$$E = \sum_{i > j} Q_{ij} \cdot x_i \cdot x_j + \sum_{i} Q_{ii} \cdot x_i$$

which shows the linear term in the QUBO matrix separated from the diagonal elements of the matrix.
The qubo solver API supports two data formats: MatrixMarket and HDF5.
Each element of the QUBO matrix must have absolute value less than or equal to 1e+20.

3.3.1.4.1. MatrixMarket format

Problem data can be prepared using the MatrixMarket format. For details about the format, see https://math.nist.gov/MatrixMarket/formats.html.

- SQBM+ supports only 'general' as 'symmetry'. However, the evaluation value returned by the qubo solver API, value, is computed using only the elements of the lower triangular matrix, including the diagonal entries. The "symmetric" specification is supported for backward compatibility, but will be deprecated in the future. Even if "symmetric" is specified, it will be processed in the same way as general.
- For specifics on how to convert a real problem to the MatrixMarket format, see the example of the MAX-CUT problem described in Example of using the qubo solver API.
- If 'integer' is specified for the data type, but the actual data is 'real', the qubo solver API does not return an error but executes computation as real-type data, prioritizing the actual data type over the specified data type.
- When specifying elements of the matrix using indices, take care not to duplicate the same elements. If problem files contain duplicates, the behavior of the qubo solver is undefined, and the system may return an error with the status code 400.

3.3.1.4.2. qubo solver API-compatible HDF5 format

Binary problem data in the HDF5 format[2]. Refer to https://www.hdfgroup.org/solutions/hdf5/ for details of the format.

- SQBM+ checks the header of an input file and recognizes its format. For this reason, the user does not need to specify the format type.
- The file structure of an HDF5 file to be accepted by SQBM+ should be one of the following two:

File structure for a sparse QUBO matrix

The CSR format is used to represent a sparse matrix. For details, see https://docs.nvidia.com/nvpl/_static/sparse/storage_format/sparse_matrix.html. The referenced ROW_INDEX corresponds to /qubo/indptr, COL_INDEX to /qubo/indices, and V to /qubo/data. The file structure is as shown below

```
+-Group("/qubo")
     +-DataSet("/qubo/data")
     |   +-Attribute["format"]="csr"
     +-DataSet("/qubo/indptr")
     +-DataSet("/qubo/indices")
```

where /qubo/data, /qubo/indptr, and /qubo/indices are one-dimensional arrays.

File structure for a dense QUBO matrix

The file structure is as shown below:

```
+-Group("/qubo")
     +-DataSet("/qubo/data")
         +-Attribute["format"]="dense"
```

where /qubo/data is a two-dimensional array compatible with a qubo matrix.

Data type

To use a file structure for a sparse QUBO matrix, use the following

```
data types for the elements of each array:
data array: float32
indices array: uint32
indptr array: uint32
```

To use a file structure for a dense QUBO matrix, use the following data

```
types for the elements of each array:
data array: float32
```

Refer to the section Example of using an HDF5 file as input data for actual usage.

3.3.1.5. Restrictions on input problem data

The problem size that can be handled by the qubo solver API has the following limitations:

- The number of qubo variables should be equal to or less than 10,000,000.
- The number of nonzero elements in the QUBO matrix is
  - Must be less than or equal to 100,000,000 for MatrixMarket format.
  - HDF5 format (file structure corresponding to sparse QUBO matrix) must be 500,000,000 or less.
  - Must be 1,000,000,000 or less in HDF5 format (file structure corresponding to dense QUBO matrix).
- HDF5 file size must be less than 2GB.
- Matrixmarket format problem data must be less than or equal to 110,000,000 lines.

However, due to memory constraints, it may not be possible to obtain a solution even if the number is less than the limit. Recommended GPU memory size are as follows:

- At least 16 GB of GPU memory is required.
- For problem data with more than 45,000 variables or 300 million interaction terms, GPU memory is required 32 GB or larger for each GPU core.

3.3.2. Response specifications of the qubo solver API

3.3.2.1. Response status

Same as the response status described in Response status common to all solvers.

3.3.2.2. Response header

Same as the response header described in Response header common to all solvers.

3.3.2.3. Response body

Same as the response body described in Response body common to all solvers, except for the properties in Table 10.

Table 10. Response body specifications of the qubo solver

Property name	Description
value	Evaluation value calculated by the QUBO model.
result	An array of qubo variables. If an array has five qubo variables, its 'result' would be `[0,1,1,0,0]` .
param	Steps and qubo solver API specific parameters algo, dt, and C.

3.3.3. Example of using the qubo solver API

This section describes how to use the qubo solver API, taking the MAX-CUT problem as an example. The MAX-CUT problem is to find the cut which maximizes the number of edges to be cut when the vertices of the graph are partitioned into two groups. In the weighted version of MAX-CUT, the edges are weighted and the problem is to find the cut with maximum weight, where the sum of the weights of the edges to be cut is maximized. The MAX-CUT problem in Figure 5 (with all weights equal to 1) has five optimal solutions.

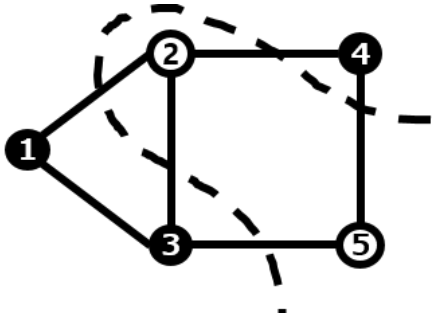

Figure 5. MAX-CUT problem

The objective function for the MAX-CUT problem can be represented by the following Ising model:

$$minimize\ E = -\frac{1}{2}\sum_{i>j} W_{ij}(1 - s_i s_j)$$

where $s_i \in \{-1, 1\}$ is a spin variable, whose value denotes which one of the two partitioned groups the vertex i belongs to, and where W_{ij} is the weight of the edge ij. If the neighboring vertices i and j, which are connected by the edge, belong to the same group, $s_i s_j = 1$; if they belong to different groups, $s_i s_j = -1$. This means that the greater the number of sets of vertices that belong to different groups and partition edges, that is, the greater the number of vertices between two vertices that are partitioned, the more the energy E will be minimized.
The inputs to the SQBM+ qubo solver API should have the QUBO format. Hence, the Ising model above must be converted to the QUBO format. Given that variables in the QUBO format are $x_i \in \{0, 1\}$, the spin variable can be expressed as $s_i = 2x_i - 1$. Substituting this into the formula for the ISING model and rearranging the resulting formula, the following objective function in the QUBO format can be obtained:

$$minimize\ E = \sum_{i>j} W_{ij}(2x_i x_j - x_i - x_j)$$

From the formula above, a QUBO matrix which will be an input to the qubo solver API can be obtained. In the example in Figure 5, the QUBO matrix will be the one like in Figure 6, where the coordinates of the linear term configure diagonal elements. Here weight W_{ij} is 1 if i and j are connected by an edge.

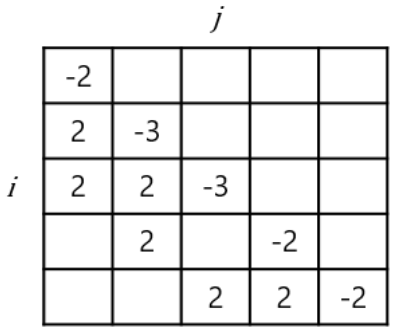

Figure 6. QUBO matrix

Below shows an input file to the qubo solver API created from the QUBO matrix in Figure 6, where qubo.txt is a file name. Note that this file is created in the MarketMatrix format.

qubo.txt

```
%%MatrixMarket matrix coordinate integer general
5 5 11
1 1 -2
2 1 2
2 2 -3
3 1 2
3 2 2
3 3 -3
4 2 2
4 4 -2
5 3 2
5 4 2
5 5 -2
```

The result of solving the MAX-CUT problem[3] with the qubo solver API using the above qubo.txt is as follows.

> $ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qubo" --data-binary "@qubo.txt"
>
> HTTP/1.1 200 OK
> X-Calculation-Time: 10.047
> X-ID: r2344244196
> Content-Type: application/json; charset=utf-8
> Content-Length: 182
> ETag: W/"b6-2RQvWpIqZ6gO70gxv9xQqMT3ZQ8"
> Date: Tue, 17 Jan 2023 09:17:48 GMT
> Connection: keep-alive
> Keep-Alive: timeout=5
>
> {"id":"r2344244196","time":10.047,"wait":0,"message":"timeout","runs":517010,"value":-5,"result":[1,1,0,0,1],"param":
> {"algo":20,"steps":2,"dt":0.7796917,"C":0.1791543},"count":97500}

Thus, the execution solution [1,1,0,0,1] is obtained. The sample results in Figure 5 show the results of optimizing the MAX-CUT problem in Figure 5. Because there are multiple optimal solutions for this example problem, the results are not always the same.

3.3.4. Example of setting parameters

From now on, examples for how to set parameters to the qubo solver API are presented.

3.3.4.1. Example of setting steps and loops

```
$ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qubo?steps=1000&loops=10" --data-binary "@qubo.txt"

HTTP/1.1 200 OK
X-Calculation-Time: 0.136
X-ID: r4641477837
Content-Type: application/json; charset=utf-8
Content-Length: 184
ETag: W/"b8-5IW9ZBtd9mXZhimG1zIj0xEQTDA"
Date: Tue, 17 Jan 2023 09:19:34 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":"r4641477837","time":0.136,"wait":0.001,"message":"finished","runs":800,"value":-5,"result":[1,1,0,0,1],"param":{"algo":20,"steps":1000,"dt":0.7796917,"C":0.1791543},"count":191}
```

3.3.4.2. Example of setting C and dt

```
$ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qubo?steps=300&loops=1&C=0.29874&dt=0.1" --data-binary "@qubo.txt"

HTTP/1.1 200 OK
X-Calculation-Time: 0.081
X-ID: r1183957498
Content-Type: application/json; charset=utf-8
Content-Length: 170
ETag: W/"aa-CraW/VGpZp4+y/lE7hl/Twt+gGc"
Date: Tue, 17 Jan 2023 09:21:42 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":"r1183957498","time":0.081,"wait":0,"message":"finished","runs":200,"value":-5,"result":[1,0,1,1,0],"param":{"algo":20,"steps":300,"dt":0.1,"C":0.29874},"count":54}
```

3.3.4.3. Example of repeating computation

The computation continues until the evaluation value ('value') specified by 'target' is obtained or until the process times out.

```
$ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qubo?steps=300&loops=0&target=-5&timeout=10" --data-binary "@qubo.txt"

HTTP/1.1 200 OK
X-Calculation-Time: 0.079
X-ID: r3501222743
Content-Type: application/json; charset=utf-8
Content-Length: 181
ETag: W/"b5-TUBZD4uvYgfePoE1iQ163oisl9w"
Date: Tue, 17 Jan 2023 09:22:40 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":"r3501222743","time":0.079,"wait":0.001,"message":"reached","runs":110,"value":-5,"result":[0,0,1,1,0],"param":{"algo":20,"steps":300,"dt":0.7796917,"C":0.1791543},"count":30}
```

3.3.4.4. Example of outputting multiple solutions

```
$ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qubo?maxout=3" --data-binary "@qubo.txt"

HTTP/1.1 200 OK
X-Calculation-Time: 10.048
X-ID: r2036897887
Content-Type: application/json; charset=utf-8
Content-Length: 410
ETag: W/"19a-VrKf6zNLBNSDbkqrDi4G+A1U97w"
Date: Tue, 17 Jan 2023 09:23:40 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":"r2036897887","time":10.048,"wait":0,"message":"timeout","runs":507080,"value":-5,"result":[0,1,0,0,1],"param":
{"algo":205,"steps":2,"dt":0.7796917,"C":0.1791543},"count":95629,"others":[{"value":-5,"result":[0,0,1,1,0],"param":
{"algo":205,"steps":2,"dt":0.7796917,"C":0.1791543},"count":95694},{"value":-5,"result":[1,1,0,0,1],"param":
{"algo":205,"steps":2,"dt":0.7796917,"C":0.1791543},"count":96091}]}
```

3.3.4.5. Example of using an HDF5 file as input data

```
$ curl -X POST -H "Content-Type: application/octet-stream" "http://sqbmplus_server:8000/solver/qubo?timeout=1" --data-binary @qubo.h5

{"id":"r4139464394","time":1.043,"wait":0,"message":"timeout","runs":70790,"value":-5,"result":[0,0,1,1,0],"param":
{"algo":20,"steps":2,"dt":0.7796917,"C":0.1791543},"count":13150}
```

The following sample program is used to convert a MatrixMarket format file to an HDF5 file.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os, itertools as it
import argparse

import numpy as np
import h5py
from scipy.sparse import csr_matrix


comp = 'gzip'
opts = 1


def triangular(M):
    '''
    Converts a matrix to a lower triangular matrix. Keep the value of xMx unchanged.

    Argument:
        M (Numpy array): Matrix

    Return:
        MM (Numpy array): Lower triangular matrix
    '''
    MM = np.zeros(M.shape)
    for i in range(M.shape[0]):
        for j in range(M.shape[1]):
            if i > j:
                MM[i,j] = M[i,j] + M[j,i]
            elif i == j:
                MM[i,j] = M[i,j]
    return MM


def parse_qubo(file):
    """
    Read Matrix Market format

    Argument:
        file: Matrix Market format
    Return:
        H (Numpy array): Matrix representing the quadratic terms of the objective function
    """
    stm = (line.strip() for line in file)

    # header
    _ = next(stm)

    n_vars, _, nnz = map(int, next(stm).split())
    H = np.zeros((n_vars, n_vars))
    for s0,s1,s2 in map(str.split, it.islice(stm, nnz)):
        H[int(s0)-1,int(s1)-1] = float(s2)

    return H


def write_matrix_HDF5(M, quboFileName, path, comp_flag, form):
    '''
    Create HDF5 file to input to SBM
     This method writes matrix data to a file.

    Argument:
        M (Numpy array): QUBO matrix
        quboFileName (String): hdf5 format file name of output destination
        path (String): A path composed of hdf5 groups and subgroups. Example "category1/category2"
        comp_flag (Boolean): compression or not. True or False
        form (String): The format that holds the matrix. "csr" or "dense" (retains all elements including 0)
    Return:
```

```
    '''
    if form == 'dense':
        with h5py.File(quboFileName, mode='a') as hf:
            group = hf.create_group(f'/{path}')
            # data type is np.float32
            if comp_flag == True:
                temp = group.create_dataset(name='data', shape=M.shape, dtype=np.float32, data=M,
                                            compression=comp, compression_opts=opts)
            else:
                temp = group.create_dataset(name='data', shape=M.shape, dtype=np.float32, data=M)
            temp.attrs['format'] = 'dense'
            hf.close()
    elif form == 'csr':
        MM = csr_matrix(M)
        # Save with the following layout
        with h5py.File(quboFileName, mode='a') as hf:
            group = hf.create_group(f'/{path}')
            # data type is np.float32
            if comp_flag == True:
                temp = group.create_dataset(name='data', shape=MM.data.shape, dtype=np.float32,
                                            compression=comp, compression_opts=opts)
            else:
                temp = group.create_dataset(name='data', shape=MM.data.shape, dtype=np.float32)
            temp[:] = MM.data[:]
            temp.attrs['format'] = 'csr'
            if comp_flag == True:
                temp = group.create_dataset(name='shape', shape=(len(M.shape),), dtype=np.int32,
                                            compression=comp, compression_opts=opts)
            else:
                temp = group.create_dataset(name='shape', shape=(len(M.shape),), dtype=np.int32)
            temp[:] = M.shape[:]
            # indices type is np.uint32
            if comp_flag == True:
                temp = group.create_dataset(name='indices', shape=MM.indices.shape, dtype=np.uint32,
                                            compression=comp, compression_opts=opts)
            else:
                temp = group.create_dataset(name='indices', shape=MM.indices.shape, dtype=np.uint32)
            temp[:] = MM.indices[:]
            # indptr type is np.uint32
            if comp_flag == True:
                temp = group.create_dataset(name='indptr', shape=MM.indptr.shape, dtype=np.uint32,
                                            compression=comp, compression_opts=opts)
            else:
                temp = group.create_dataset(name='indptr', shape=MM.indptr.shape, dtype=np.uint32)
            temp[:] = MM.indptr[:]
            hf.close()


def main():
    argparser = argparse.ArgumentParser()
    argparser.add_argument("input_file_path", type=str)
    argparser.add_argument("--input_type", type=str, choices=['qplib', 'qubo'], default='qplib')
    argparser.add_argument("--format", type=str, choices=['csr', 'dense'], default='csr')
    argparser.add_argument("--comp", action='store_true')
    args = argparser.parse_args()
    comp = args.comp
    form = args.format

    # Create output file name
    REQ_HDF5 = os.path.splitext(os.path.basename(args.input_file_path))[0] + '.h5'
    print('output:', REQ_HDF5)
    if os.path.exists(REQ_HDF5):
        os.remove(REQ_HDF5)

    with open(args.input_file_path) as f:
        if args.input_type == 'qplib':
            (_, _, p_sense, H_o, g_o, f_o, A_o, cl_o, cu_o) = parse_qplib(f)
```

```
            n = H_o.shape[0]
            m = A_o.shape[0]
            if (p_sense.lower() == "maximize"):
                H, g = -H_o, -g_o
            else:
                H, g = +H_o, +g_o

            write_matrix_HDF5(triangular(H), REQ_HDF5, '/qubo', comp, form)
            write_vector_HDF5(g, REQ_HDF5, '/linear', comp)
            # Group constraints in one directory
            write_matrix_HDF5(A_o, REQ_HDF5, '/constraints/coeff', comp, form)
            write_vector_HDF5(cl_o, REQ_HDF5, '/constraints/lower', comp)
            write_vector_HDF5(cu_o, REQ_HDF5, '/constraints/upper', comp)

        else:
            H = parse_qubo(f)
            write_matrix_HDF5(H, REQ_HDF5, '/qubo', comp, form)

if __name__ == '__main__':
    main()
```

3.4. qplib solver API

The qplib solver API solves a quadratic programming problem consisting of binary variables, a quadratic function as the objective function, and linear constraints. Aside from the QUBO coefficient matrix, which is the objective function, the qplib solver API receives linear constraints and returns the best solution found.

3.4.1. Definition of a problem (quadratic programming problem)

- variable (x is the decision variable vector, x_i is the element):

$$x_i \in \mathbb{R} \; or \; x_i \in \{0, 1\}$$

- parameter:

$$Q, A, LHS, RHS$$

- constrained quadratic programming problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2} x^T \cdot Q \cdot x + B \cdot x \\ &\text{subject to } LHS \leq A \cdot x \leq RHS \end{aligned}$$

Q, which is used in the objective function, is a QUBO matrix, while B is a coefficient vector of a linear term.
A, which is used in linear constraint conditions, is an $M \times N$ matrix. M is the number of constraints, whereas N is the number of variables.

- $-1 \leq x_1 + x_2 + x_3 \leq 1$
- $-3 \leq x_1 + x_3 \leq 1$
- $-4 \leq x_1 \leq 1$

Based on the matrix described above, multiple linear constraints like the three right above are combined and expressed as $LHS \leq A \cdot x \leq RHS$.
In the example above, vectors expressed as $LHS = \{-1, -3, -4\}$, $RHS = \{1, 1, 1\}$ are obtained.

- Here, $LHS and RHS$ must satisfy one of the following constraints, where the m-th component of an LHS vector is expressed as LHS_m:
  - Inequality constraints with the upper and lower boundaries
    $LHS_m < RHS_m,\ LHS_m \in \mathbb{R},\ RHS_m \in \mathbb{R}$
  - Inequality constraints without the upper boundaries
    $LHS_m \in \mathbb{R}, RHS_m = +\inf$
  - Inequality constraints without the lower boundaries
    $LHS_m = -\inf, RHS_m \in \mathbb{R}$
  - Equality constraints
    $LHS_m = RHS_m,\ LHS_m \in \mathbb{R},\ RHS_m \in \mathbb{R}$

QUBO matrix, each element of B vector must have absolute value less than or equal to 1e+20.
If the absolute value of each element of the A matrix, LHS, and RHS vectors exceeds 1e+20, scale while maintaining the constraint so that each element of the A matrix, LHS, and RHS does not exceed 1e+20.

3.4.2. Request specifications of the qplib solver API

3.4.2.1. HTTP request

http://{ip}:{port}/solver/{solver}?{parameter}
select qplib as {solver}.

3.4.2.2. Request header

Table 11. Request header field and a variable that need to be set

Header field	Description
Content-Type	application/octet-stream

3.4.2.3. Request parameters

In addition to the parameters available in qubo solver API, the following parameters are also available.

Table 12. qplib solver API specific parameters

Parameter	Default	Description
PD3Orate	0	Parameter that determines the number of PD3O algorithm execution steps. SQBM+ searches for solutions using the SB algorithm, but if PD3Orate is set to anything other than 0, it also searches for solutions using another algorithm (called the PD3O algorithm). For the steps of the SB algorithm, the PD3O algorithm performs PD3Orate*steps times to update decision variables.
phi	0	Parameter that determines the behavior of the PD3O algorithm. Specify a real number between 0 and 1. If it is close to 0, it behaves like the SB algorithm. If it is close to 1, it narrows the search range of solutions, but it may find a better solution than the SB algorithm.
detail_level	0	Parameter that determines the level of the response information details when feasible solution is not found. Specify 0 or 1. If select 1 and feasible solution is not found, SQBM+ return additional details for "no result".
detail_log	0	Parameter that determines the level of detail of the logging information when feasible solution is not found. Specify 0 or 1. If select 1 and feasible solution is not found, SQBM+ write value of each constraint vioration rate to log file.

3.4.2.4. Problem data specification

In the problem data specification, an HDF5 format file and a qplib format file can be used.

3.4.2.4.1. HDF5 format

An HDF5 file that the qplib solver API can use should be configured as below.
Specify 3.40283e+38 when specifying inf.

Table 13. HDF5 file format used by the qplib solver API

Structure	Group name or the content
/qubo	`Q` matrix data
/linear	`B` vector data
/constraints	group name for the constraint conditions
/coeff	`A` matrix data
/lower	`LHS` vector data
/upper	`RHS` vector data

Each data set in an HDF5 file should follow the rules below:

Table 14. Attribute settings for each data type

Data	Data type	Content attributes
vector	array	Define a vector as a one-dimensional array. The data type of each component is float32. For the attribute, set the `"format" to "dense"`.
sparse matrix	Sparse array	Same as qubo solver API-compatible HDF5 format. For the attribute, set the `"format" to "csr"`.
dense matrix	Dense array	Same as qubo solver API-compatible HDF5 format. For the attribute, set the `"format" to "dense"`.
`Q` matrix data	Dense array Sparse array	To set data as a dense matrix, set it as the data of the Dense array data type. To set data as a sparse matrix, set it as the data of the Sparse array data type.
`B` vector	array	Set the `"format" to "dense"`.
`A` matrix data	Dense array Sparse array	To set data as a dense matrix, set it as the data of the Dense array data type. To set data as a sparse matrix, set it as the data of the Sparse array data type.
`LHS` vector	array	Set the `"format" to "dense"`.
`RHS` vector	array	Set the `"format" to "dense"`.

3.4.2.4.2. qplib format

The qplib solver API uses a format that sets information by tag specification equivalent to the information set by QPLIB[4]. This format is hereafter referred to as the qplib format. See http://qplib.zib.de/doc.html for more information about QPLIB. The qplib format categorizes problem types by objective function, variable type, and constraint type. For problem types, specify a total of three types: Objective type, Variables type, and Constraints type. Each of these is specified using one alphabet. For available types and the details about specified strings, see http://qplib.zib.de/doc.html. The qplib format is also used by the pubo solver API, so this chapter also contains information on using it with the pubo solver API. In a qplib format file, enter the following three items in the file header:

- Problem name: string that starts with "QPLIB_".
- Problem type settings:
  - qplib solver API: Select the problem type from ['QBB', 'QBL', 'QCB', 'QCL', 'QGB', 'QGL'].
  - pubo solver API: Select the problem type from ['QBB','QCB', 'QGB'].
- To obtain the minimum value of the objective function, specify "minimize"; to obtain the maximum value, specify "maximize".

Next, enter the problem settings in the file using the tags described in Table 15.
When specifying inf, specify 1.79769313486232e+308.

Table 15. Setting tag information

tag	description
number of variables	Specify the number of variables to be used. For the qplib solver API, specify an integer between 2 and 10,000,000. For the pubo solver API, specify an integer between 2 and 100,000. Example 3 # number of variables
number of constraints	Specify the number of linear constraint conditions to be used. Note that with the Constraints type B (without linear constraint conditions), specification of a number other than 0 (zero) results in an error. Pubo solver is not available. Specify an integer between 0 and 1,000,000. Example 3 # number of constraints
default left-hand-side value	The settings for the lower limit value of all linear constraint conditions are made for this tag. Specify the default configuration value for all the components of an LHS vector. With the Constraints type B (without linear constraint conditions), do not use this tag. Pubo solver is not available. Specify a real number. Constraints type Example 0 # default left-hand-side value
default right-hand-side value	The settings for the upper limit value of all linear constraint conditions are made for this tag. Specify the default configuration value for all the components of an RHS vector. With the Constraints type B (without linear constraint conditions), do not use this type. Pubo solver is not available. Specify a real number. Example 0 # default right-hand-side value
default value for linear coefficients in objective	Specify the default configuration value of all the components of a B vector. Specify a real number whose absolute value is 1e+20 or less. Example 0 # default value for linear coefficients in objective
default variable type	Specify the default data type of each variable. With Variables type B (binary) or C (continuous variable), do not use this tag. Select one of the following: 0: continuous real number 1: binary Example 1 # default variable type
default variable upper bound value	Specify the domain's upper limit value of all the variables. With the Variables type B (binary), do not use this tag. Specify a real number greater than or equal to -1e+20. Example 1.5 # default variable upper bound value

tag	description
default variable lower bound value	Specify the domain's lower limit value of all the variables. With the Variables type B (binary), do not use this tag. Specify a real number less than or equal to 1e+20. Example 1.5 # default variable lower bound value
number of quadratic terms in objective	The settings for a QUBO matrix are made for this tag. Specify an integer of 1 or more as the setting value of the tag, and specify the setting value of each matrix component for that integer. Note that the maximum number of nonzero elements in a matrix is no more than 100,000,000 for the qplib solver API and no more than 10,000,000 for the pubo solver API. The matrix components for which no settings are made here are set to 0 (zero). Specify a real number whose absolute value is 1e+20 or less. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. Specify variable indices in descending order. number of non-zero Q_{ij}'s # number of quadratic terms in objective [variable index] [variable index] [coefficient value] [variable index] [variable index] [coefficient value] …… Repeat above as many times as the number of non-zero Q_{ij}'s …… [variable index] [variable index] [coefficient value] Example 2 # number of quadratic terms in objective 2 1 -2 3 2 3.5
number of pubo terms in objective	Specify terms of cubic or higher. Specify an integer of 1 or more as the setting value of the tag, and specify the setting value of each matrix component for that integer. However, the number of elements that can be set must be 10,000,000 or less together with the number of quadratic terms in objective. This tag is only available for the pubo solver API. Specify a real number whose absolute value is 1e+20 or less. Specify as follows. Variable indices start from 1 and are numbered sequentially from the top of the variable vector. Number of nonzero # number of quadratic terms in objective [variable index] ... number of degree ... [variable index] [coefficient value] [variable index] ... number of degree ... [variable index] [coefficient value] …… Repeat above as many times as the number of non-zero T_{ijk} and T_{ijkl} …… [variable index] ... number of degree ... [variable index] [coefficient value] Example 2 # number of pubo terms in objective 2 1 1 -2 3 2 2 3.5
number of non-default linear coefficients in objective	The B vector that has been set in the "default value for linear coefficients in objective" is overwritten. Specify the settings value of components of the vector to overwrite. For those components for which no settings are made here, the value that has been set to the "default value for linear coefficients in objective" tag is used. Specify a real number whose absolute value is 1e+20 or less. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of components of the B vector to overwrite # number of non-default linear coefficients in objective [variable index] [coefficient value] [variable index] [coefficient value] …… Repeat above as many times as the number of components. …… [variable index] [coefficient value] Example 2 # number of non-default linear coefficients in objective 1 -2 2 3.5
number of linear terms in all constraints	The settings for an A matrix are made for this tag. Specify the settings value of each matrix component. Note that an A matrix is a matrix in which the value specified in the "number of constraints" tag corresponds to the length of the row, and the number of variables, to the length of the column. Pubo solver is not available. The matrix components for which no settings are made here are set to 0 (zero). Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable

tag	description
	vector to the bottom. number of non-zero A_{ij}'s # number of linear terms in all constraints [variable index] [variable index] [coefficient value] [variable index] [variable index] [coefficient value] …… Repeat above as many times as the number of non-zero A_{ij}'s …… [variable index] [variable index] [coefficient value] Example 2 # number of linear terms in all constraints 2 1 -2 3 2 3.5
number of non-default left-hand-sides	The LHS vector that has been set in the "default left-hand-side value" is overwritten. Specify the settings value of components of the vector to overwrite. For those components for which no settings are made here, the value that has been set to the "default left-hand-side value" tag is used. Pubo solver is not available. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of the LHS vector to overwrite # number of non-default left-hand-sides [variable index] [lower limit value of the constraint conditions] [variable index] [lower limit value of the] …… Repeat above as many times as the number of components. …… [variable index][lower limit value of the constraint conditions] Example 2 # number of non-default left-hand-sides 1 -2 2 3.5
number of non-default right-hand-sides	The RHS vector that has been set in the "default right-hand-side value" is overwritten. Specify the settings value of components of the vector to overwrite. For those components for which no settings are made here, the value that has been set to the "default right-hand-side value" tag is used. Pubo solver is not available. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of the RHS vector to overwrite # number of non-default right-hand-sides [variable index] [upper limit value of the constraint conditions] [variable index] [upper limit value of the constraint conditions] …… Repeat above as many times as the number of components. …… [variable index] [upper limit value of the constraint conditions] Example 2 # number of non-default right-hand-sides 1 -2 2 3.5
number of non-default variable upper bounds	The domain's upper limit value of each variable that has been set to the "default variable upper bound value" is overwritten. Specify a value equal to or greater than -1e+20 for the domain's upper limit value of the variable to overwrite. For those components for which no settings are made here, the value that has been set to the "default variable upper bound value" tag is used. With the Variables type B (binary), do not use this tag. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of the domain's upper limit values to overwrite # number of non-default variable upper bounds [variable index] [upper limit value] [variable index] [upper limit value] …… Repeat above as many times as the number of the domain's upper limit values. …… [variable index] [upper limit value] Example 2 # number of non-default variable upper bounds 1 -2 2 3.5

tag	description
number of non-default variable lower bounds	The domain's lower limit value of each variable that has been set to the "default variable lower bound value" is overwritten. Specify a value equal to or less than 1e+20 for the domain's lower limit value of the variable to overwrite. For those components for which no settings are made here, the value that has been set to the "default variable lower bound value" tag is used. With the Variables type B (binary), do not use this tag. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of the domain's upper limit values to overwrite # number of non-default variable lower bounds [variable index] [lower limit value] [variable index] [lower limit value] …… Repeat above as many times as the number of the domain's lower limit values. …… [variable index] [lower limit value] Example 2 # number of non-default variable lower bounds 1 -2 2 3.5
number of non-default variable types	The data types of variables that have been set in the "default variable type" is overwritten. Specify the data types of the variables to overwrite. For those data types of variables for which no settings are made here, the value that has been set to the "default variable type" tag is used. With the Variables type B (binary) or C (continuous variable), do not use this tag. Make the following settings. Here the variable index starts with 1 and is numbered consecutively from the top of the variable vector to the bottom. number of data types of variables to overwrite. # number of non-default variable types [variable index] [data type] [variable index] [data type] …… Repeat above as many times as the number of data types. …… [variable index] [data type] Example 2 # number of non-default variable types 1 0 2 0

3.4.2.5. Restrictions on input problem data

The problem size that can be handled by the qplib solver API has the following limitations:

- The number of variables should be equal to or less than 10,000,000.
- The number of nonzero elements in the matrix is
  - Must be less than or equal to 100,000,000 for qplib format.
  - Must be less than or equal to 500,000,000 in HDF5 format (Q matrix is sparse).
  - Must be less than or equal to 800,000,000 in HDF5 format (where the Q matrix is a dense matrix).
- The number of constraints should be equal to or less than 1,000,000.
- HDF5 file size must be less than 2GB.
- qplib format problem data must be less than or equal to 110,000,000 lines.

3.4.3. Response specifications of the qplib solver API

3.4.3.1. Response status

Same as the response status described in Response status common to all solvers.

3.4.3.2. Response header

Same as the response header described in Response header common to all solvers.

3.4.3.3. Response body

Same as the response body described in Response body common to all solvers, except for the properties in Table 16.

Table 16. Response body specifications of the qplib solver API

Property name	Description
value	Evaluation value calculated by the objective function of a quadratic programming problem.
result	An array of variables. If an array has five variables, its "result" would be `[0,1,1,0,0]` .
param	Displays qplib solver API-specific parameters, including algo, steps, dt, and C.

3.4.3.4. Error messages

Same as the messages described in Error messages, except for the properties in Table 17.

Table 17. List of error messages

Status code	Message	Description
400	Refer Table.18	If set parameter `"detail_level"` to 1 and no feasible solution found, qplib solver response specific response. The response format is JSON. The Table.18 describes property format.

Table 18. `"no result"` response body specifications of the qplib solver API

Property name	Description
main_info	Main message.
message	Elements of "main_info". Error message as a string.
additional	Additional result information objects.
runs	Elements of "additional". Number of calculated executions.
additional_info	Elements of "additional". An array of up to 10 of the most recently calculated results.
value	Elements of "additional_info". Evaluation value computed based on the solutions represented by 'result' in this table. example, `-3`
result	Elements of "additional_info". An array for each variable. example, `[0,0,1,1,0]`
param	Values of parameter settings used when the best solution is obtained. JSON format. Parameters displayed vary depending on which solver and algorithm to use. example, `{"algo": 20,"steps": 334,"dt": 1,"C": 0.5}`
violated_condition	Elements of "additional_info". The constraint number indicating which constraint is violated (origin number 0) is the order of constraints set by the input data. example, `[0,1]`
violation_rate	Elements of "additional". The rate at which each constraint is violated. The elements of the array are arranged in the order of constraints set by the input file. example, `[0.1,1]`

3.4.4. Example of using the qplib solver API

3.4.4.1. Sample problem description (knapsack problem)

Taking a 0-1 knapsack problem as an example, this section describes how to use the qplib solver API. A 0-1 knapsack problem can be formulated as:

$$\max_x \sum_i v_i x_i$$
$$s.t. \sum_i w_i x_i \leq W$$

where v_i is the value of the item i, x_i determines whether the item i is selected (1) for each binary variable or not (0), w_i is the weight of the item i, and W is the upper limit value of the net weight of the selected items.

In this sample problem, let us set v, w, and W as follows:

$$v = \begin{bmatrix} 11 & 13 & 17 & 19 \end{bmatrix}$$
$$w = \begin{bmatrix} 2 & 3 & 5 & 7 \end{bmatrix}$$
$$W = 10$$

With the settings above, this knapsack problem can be transformed into the following quadratic programming problem:

$$\min_x \frac{1}{2} x^{\mathrm{T}} Q x$$
$$s.t.\ LHS \leq Ax \leq RHS$$

where

$$Q = -\begin{bmatrix} 22 & 0 & 0 & 0 \\ 0 & 26 & 0 & 0 \\ 0 & 0 & 34 & 0 \\ 0 & 0 & 0 & 38 \end{bmatrix}$$
$$A = \begin{bmatrix} 2 & 3 & 5 & 7 \end{bmatrix}$$
$$LHS = 0$$
$$RHS = 10$$

3.4.4.2. Sample qplib format file

Converting the sample problem in the previous section into qplib format, we would obtain the following:

knapsack.qplib

```
QPLIB_knapsack
QBL
minimize
4 # number of variables
1 # number of constraints
0 # default left-hand-side value
10 # default right-hand-side value
0 # default value for linear coefficients in objective
4 # number of quadratic terms in objective
1 1 -22
2 2 -26
3 3 -34
4 4 -38
4 # number of linear terms in all constraints
1 1 2
1 2 3
1 3 5
1 4 7
```

Execution of the problem above results in the following:

```
$ curl -i -H "Content-Type: application/octet-stream" -X POST "http://sqbmplus_server:8000/solver/qplib" --data-binary @knapsack.qplib

HTTP/1.1 200 OK
X-Calculation-Time: 10.014
X-ID: r3552467021
Content-Type: application/json; charset=utf-8
```

Content-Length: 180
ETag: W/"b4-gY2unru0uk3uTgzDl2EozRSOLo0"
Date: Tue, 17 Jan 2023 10:14:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":"r3552467021","time":10.014,"wait":0.001,"message":"timeout","runs":494260,"value":-60,"result":[1,1,1,1],"param":{"algo":205,"steps":2,"dt":1,"C":0.01130918},"count":101495}

The solution obtained is "$x = [1, 1, 1, 1]$".

3.4.5. Example of setting parameters

Same as Example of setting parameters.

3.5. pubo solver API

The pubo solver API performs optimal search for combinatorial optimization problems involving up to fourth order terms.

3.5.1. Definition of the problem (a problem with higher order terms up to order 4)

- variables (x is the decision variable vector, x_i is the element):

$$x_i \in \mathbb{R} \; or \; x_i \in \{0, 1\}$$

- Parameters:

$$T_{ijkl}, T_{ijk}, Q_{ij}, B_i$$

- Objective function:

$$\text{minimize } \frac{1}{4!}\sum_{ijkl} T_{ijkl}x_i x_j x_k x_l + \frac{1}{3!}\sum_{ijk} T_{ijk}x_i x_j x_k + \frac{1}{2!}\sum_{ij} Q_{ij}x_i x_j + \sum_i B_i x_i$$

$T_{ijkl}, T_{ijk}, Q_{ij}$, and B_i used in the objective function are the 4th order interaction coefficient of the optimization problem, the 3rd order interaction coefficient of the optimization problem, and the optimization problem is the second-order interaction coefficient of , the linear term coefficient of the optimization problem.

Each element of $T_{ijkl}, T_{ijk}, Q_{ij}, B_i$ must be less than or equal to 1e+20 in absolute value.

3.5.2. Request specifications of the pubo solver API

3.5.2.1. HTTP requests

http://{ip}:{port}/solver/{solver}?{parameter}
, select pubo as the {solver}.

3.5.2.2. Request header

Table 19. Request header to be set and setting values

Header field	Description
Content-Type	application/octet-stream

3.5.2.3. Request parameters

Same parameters as available in qubo solver API.

3.5.2.4. Problem data specification

Problem data specifications can use qplib format files. For details, refer to qplib format.

3.5.2.5. Restrictions on input problem data

The problem size that the pubo solver API can handle has the following limitations.

- The number of variables must be less than or equal to 100,000.
- The number of nonzero elements in the matrix must be less than or equal to 10,000,000.
- The problem data must be less than or equal to 110,000,000 lines.

However, due to memory constraints, it may not be possible to obtain a solution even if the number is less than the limit. The recommended memory size is follows.

- CPU memory: 60 GB or more
- GPU memory: 16 GB or more.

Also, depending on the combination of the number of variables and the number of non-zero elements in the matrix, pre-calculation processing may take time.

3.5.3. Response specifications of the pubo solver API

3.5.3.1. Response status

Same as the response status described in Response status common to all solvers.

3.5.3.2. Response header

Same as the response header described in Response header common to all solvers.

3.5.3.3. Response body

Same as the response body described in Response body common to all solvers, except for the properties in Table 20.

Table 20. Response body specification of pubo solver API

Property name	Description
value	Evaluation value calculated by the objective function.
result	An array of variables. If an array has five variables, its "result" would be `[0,1,1,0,0]` .
param	Displays qplib solver API-specific parameters, including algo, steps, dt, and C.

3.5.4. Examples of using the pubo solver API

3.5.4.1. Sample problem description (MAX-3SAT problem)

An example of the use of the pubo solver API is its application to the maximization problem of satisfaction (hereafter the MAX-SAT problem). The MAX-SAT problem is the following problem.

Considering the boolean variables $x_1, x_2, x_3, \cdots$, these boolean variables, or $(x_1 \vee \neg x_2 \vee x_3)$ which takes the OR operation of the negation of the boolean variables. You can think of logical formulas such as This is called a clause. The MAX-SAT problem is a problem to find the maximum number of clauses that can be true among multiple clauses to be satisfied. When the clause consists of two boolean variables, it is called a MAX-2SAT problem, and when it consists of three boolean variables, it is called a MAX-3SAT problem.

Consider the MAX-3SAT problem as an example. Here boolean variables x_1, x_2, x_3, x_4, x_5, clauses $(x_1 \vee \neg x_2 \vee x_3)$, $(\neg x_1 \vee x_2 \vee x_3)$, $(x_3 \vee x_4 \vee x_5)$.
The following procedure converts the clause into a polynomial and converts it into an objective function that should minimize the MAX-3SAT problem.
(1) Convert a boolean variable to (1-x), with x as a binary variable. Negating a boolean variable converts it to x. A binary variable takes 1 when the boolean variable is true.
(2) Convert OR operation to product.
(3) Execute the transformations (1) and (2) for all clauses and take the sum.
By the above procedure the MAX-3SAT problem is $(1-x_1)x_2(1-x_3) + x_1(1-x_2)(1-x_3) + (1-x_3)(1-x_4)(1-x_5)$ can be converted to If you organize this, it will be $1 + x_1 + x_2 - x_3 - x_4 - x_5 - 2x_1x_2 - x_1x_3 - x_2x_3 + x_3x_4 + x_3x_5 + x_4x_5 + 2x_1x_2x_3 - x_3x_4x_5$. Minimizing this, the constant 1 is not needed as an objective function and can be removed. Finding the decision variable that minimizes the objective function gives the value of the boolean variable that is the solution to the original MAX-3SAT problem.

3.5.4.2. Sample qplib format file

The qplib format file for the sample problem dealt with in the previous section looks like this:
The objective function handled by the pubo solver API has a coefficient such as (1/2!) in front of each order term, so be careful about the value you enter.

max3sat.qplib

```
QPLIB_max3sat
QBB
minimize
5 # number of variables
0 # default value for linear coefficients in objective
6 # number of quadratic terms in objective
2 1 -4
3 1 -2
3 2 -2
4 3 2
5 3 2
5 4 2
2 # number of pubo terms in objective
3 2 1 12
5 4 3 -6
5 # number of non-default linear coefficients in objective
1 1
2 1
3 -1
4 -1
5 -1
```

The result of running the above problem is:

> $ curl -i -H "Content-Type: application/octet-stream" -X POST "http://{sqbmplus_server}:8000/solver/pubo?timeout=1" --data-binary @max3sat.qplib
>
> HTTP/1.1 200 OK
> X-Calculation-Time: 1.033
> X-ID: r3920556539
> Content-Type: application/json; charset=utf-8
> Content-Length: 184
> ETag: W/"b8-r2bdD8hfoN77bHLsEM051yMpi5c"
> Date: Mon, 15 May 2023 05:49:13 GMT
> Connection: keep-alive
> Keep-Alive: timeout=5
>
> {"id":"r3920556539","time":1.033,"wait":0.001,"message":"timeout","runs":57860,"value":-1,"result":[0,1,1,0,1],"param":{"algo":204,"steps":2,"dt":0.9872935,"C":0.1727053},"count":2405}

The resulting solution is $x = [0, 1, 1, 0, 1]$.
From this result the boolean variables x_2, x_3, and x_5 are true and x_1 and x_4 are false.

3.6. How to use the computation API as a sampler

The computation API can obtain not only solutions near the optimal solution, but also multiple solutions that follow the Boltzmann distribution with the objective function as the energy. SQBM+ users can use these solutions to compute statistical approximations such as means assuming a Boltzmann distribution.
This function is used by specifying the entry point of the qubo solver API, qplib solver API, or pubo solver API according to the objective function. For example, if it is a QUBO problem, specify the entry point of the qubo solver API.
This function samples the solution according to the probability distribution of $\frac{1}{Z}exp(-\beta H)$ with the objective function H. β is a hyperparameter proportional to the inverse of temperature. Query parameters C and β have the relationship $\beta \simeq C$.
In addition to specifying the solver, this function requires specifying the required number of samples. Specify the number of samples through the query parameter (loops). For loops, see Request parameters.
This function needs to find a solution that is not constrained near the optimal solution. To switch to such control of solver processing, flags and additional information can be specified in the query parameters. The required parameters are:

- loops: set number of samples to seek
- steps: set a value between 1000 and 10000
- maxout: value sufficiently larger than loops
- timeout: set sufficient time to obtain the number of solutions specified by maxout
- C: set a fixed value. Equivalent to the reciprocal of temperature β in the sampler
- dt: around 0.01 / ($\bar{Q} \times C$). $\bar{Q}$ is the representative value of the matrix element size.

In addition to the above, the following must be set as query parameters to use the sampler function.

- HMCweight: set to 1
- HMCsteps: Set a value between 10 and 100

Constraints:

- steps should be increased as $dt \times HMCsteps$ becomes smaller.
- If $dt \times \sqrt{C}$ and dt are too large, the deviation from the expected distribution becomes large.
- If C is too small, the probability distribution approaches white noise.
- Specify algo=15,20 if the decision variable is a binary variable, and specify algo=15 if it is a continuous variable.

4. Health check API

The health check API within the SQBM+ computation API returns the status of the API.

4.1. Request specifications

4.1.1. HTTP request

- HTTP method to use: GET
- URL configuration: http://{ip}:{port}/healthcheck
  - ip: IP address of the machine on which the SQBM+ computation API is operating.
  - port: port number of the SQBM+ server

4.1.2. Request header

No settings are required.

4.1.3. Request parameters

No settings are required.

4.2. Response specifications

4.2.1. Response status

If the SQBM+ computation API is working properly, the status code 200 is returned, if not, 503 is returned.
Connection refused if SQBM+ itself is down.

4.2.2. Response header

Table 21. Response header field and a value to be set

Header field	Description
Content-Type	application/health+json

4.2.3. Response body

The HTTP response body shows the server status.

Table 22. List of response messages

Status code	Message	Description
200	pass	The SQBM+ computation API is working properly.
503	fail	The SQBM+ computation API is stopped.

4.3. Example of using the Health check API

```
$ curl -i -X GET "http://sqbmplus_server:8000/healthcheck"
HTTP/1.1 200 OK
Content-Type: application/health+json; charset=utf-8
Content-Length: 17
ETag: W/"11-pEowaKzwbXVLfeOv1IndFqAk+sg"
Date: Thu, 12 Oct 2023 11:12:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"status":"pass"}
```

5. Version check API

The Version API within the SQBM+ computation API returns the version of the SQBM+ you are using.

5.1. Request specifications

5.1.1. HTTP request

- HTTP method to use: GET
- URL configuration: http://{ip}:{port}/version
  - ip: IP address of the machine on which the SQBM+ computation API is operating.
  - port: port number of the SQBM+ server

5.1.2. Request header

No settings are required.

5.1.3. Request parameters

No settings are required.

5.2. Response specifications

5.2.1. Response body

The HTTP response body shows the version of SQBM+ (Table 39).

Table 23. Response body specifications of the Version API

Property name	Description
version	version name

5.3. Example of using the Version API

```
$ curl -i -X GET "http://sqbmplus_server:8000/version"
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 19
ETag: W/"13-VguZll+apkiTX3mAkWiJBggcQtU"
Date: Mon, 16 Oct 2023 04:39:20 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"version":"2.0.3"}
```

6. System configuration parameters

System configuration parameters listed in Table 22 can be set in the /home/{user directory}/config file within the SQBM+ server. Usually there is no need to change these values. If you do change the values, reboot the SQBM+ server instance for the changes to take effect in the SQBM+ API. {user directory} is :

- ec2-user for AWS.
- azureuser for Azure.
- sbm-user for other than the above.

Table 24. System configuration parameters

Parameter	default	Description
server_port	8000	Port number of the SQBM+ server
payload_limit	2000000000	Maximum size of the request body in bytes. The default is 2GB. If multiple requests that exceed the default configuration value are issued, computational operations may abend due to memory problems, which may result in disconnection in communication. If this is the case, reduce the number_of_workers to be described later. Additionally, it is recommended to issue requests one at a time.
timeout_upper_bound	3600	Upper computation time limit in seconds. Regardless of the value of the timeout parameter, computation ends when the upper time limit is reached.
number_of_workers	4	Number of worker threads in the SQBM+ server that accepts computation requests.
max_requests	20	Number of computation requests acceptable at one time. The SQBM+ computation API accepts multiple requests concurrently but calculates them one by one; currently it does not process them in parallel. For computation requests with sizes more than hundreds of MB, consider issuing those requests one at a time to avoid a server error.
default_timeout	10	Default value of the computation parameter 'timeout'.
default_maxwait	60	Default value of the computation parameter 'maxwait'.
default_maxout	1	Default value of the computation parameter 'maxout'. Specify an integer greater than or equal to 1.

7. Maintenance and security

7.1. Log files

Log files are created under the /var/log/sqbm-plus/ directory in the SQBM+ server. To obtain log files, collect them by issuing the scp command from a machine other than the SQBM+ server.
[important] After collect log files, you must reboot the SQBM+ server instance.

Below is an example[5] of collecting all the log files and saving them as logfiles.tgz. If the client instance is using OpenSSH 8.7 or a later version, use the option -O with the scp command.

```
$ scp -i private-key.pem -r sqbmplus_server:/var/log/sqbm-plus/ SBM_logs
$ ls SBM_logs
main.log  sbm-engine.log  start.log
$ tar zcf logfiles.tgz SBM_logs
$ tar tf logfiles.tgz
SBM_logs/
SBM_logs/start.log
SBM_logs/sbm-engine.log
SBM_logs/main.log
```

7.1.1. Log rotation

7.1.1.1. Conditions for Log Rotation

When the file size reaches 10MB, log rotation will be performed.

7.1.1.2. Log rotation behavior

Change the current log file to ***.1. Each past log file increments the last number in the file name by one.
Example: If the log file name is main.log, it will rotate as follows and main.log will be an empty file.

```
---------------------------------
  Before rotation    After rotation
---------------------------------
  main.log -> main.log.1
  main.log.1 -> main.log.2
  main.log.2 -> main.log.3
      :             :
  main.log.9 -> main.log.10
---------------------------------
```

A maximum of 99 generations of log files can be managed. For main.log, there are 100 files, main.log, main.log.1, ..., main.log.99. The file that was main.log.99 before log rotation is deleted after rotation.

8. Troubleshooting

8.1. Actions when the SQBM+ stops

The SQBM+ API automatically recovers after it stops. If the API abends, check the API status after a few seconds using the health check API.
Sometimes SQBM+ stops because there is not enough memory to process requests. If there is no response to the request, please check /var/log/sqbm-plus/main.log, and if there is ERROR a worker died, null, SIGTERM, there is a possibility that the CPU memory is insufficient. Consider the scaling up of CPU memory. If you see error, [500,"loading error: 3"] in the log, it corresponds to insufficient GPU memory. Consider the scaling up of GPU memory.
If the health check API does not respond or the message of the response body is "fail", follow the procedure in Log files to collect log files and reboot the SQBM+ server instance.
There may be a case where there is no problem with the health check API but the SQBM+ computation API still does not respond. If so, do the same as above. That is, follow the procedure in Log files to collect log files and reboot the SQBM+ server instance.
If the problem is not resolved even after the reboot and you need to contact support for help, provide them with your log files.

8.2. Actions when the SQBM+ does not response

When solving large problems, it may take some time to receive a response from SQBM+. Even you set the timeout, SQBM+ may take up to 1 minute beyond the specified timeout to respond. When SQBM+ take too long time for you to wait, then stop computation API and reboot the SQBM+ server instance.

9. Disclaimer

THE SOFTWARE AND SERVICES ARE PROVIDED "AS IS." YOU AND YOUR USERS ACCEPT AND ASSUME THE ENTIRE RISK AS TO THE QUALITY, PERFORMANCE AND RESULTS OF ACCESS OR USE OF THE SOFTWARE AND SERVICES. TOSHIBA DIGITAL SOLUTIONS CORPORATION (TDSL) AND TDSL'S AFFILIATES AND LICENSORS MAKE NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE REGARDING THE SOFTWARE AND SERVICES, INCLUDING ANY WARRANTY THAT THE SOFTWARE AND SERVICES WILL BE UNINTERRUPTED, ERROR FREE OR FREE OF HARMFUL COMPONENTS, OR THAT YOUR CONTENT AND DATA, WILL BE SECURE OR NOT OTHERWISE LOST OR DAMAGED. EXCEPT TO THE EXTENT PROHIBITED BY LAW, TDSL AND TDSL'S AFFILIATES AND LICENSORS DISCLAIM ALL WARRANTIES, INCLUDING ANY IMPLIED WARRANTIES OF PERFORMANCE, MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR QUIET ENJOYMENT, AND ANY WARRANTIES ARISING OUT OF ANY COURSE OF DEALING OR USAGE OF TRADE.

User's Manual for SQBM+™ V2

August 5, 2024
A3 edition
MWS5551A

Published by
72-34 Horikawa-cho, Saiwai-ku Kawasaki-shi, Kanagawa 212-8585, Japan

1. However, it does not guarantee optimal solutions. ↩
2. HDF is a registered trademark of The HDF Group. ↩
3. In this example and all other examples that follow, substitute sqbmplus_server with the host name of the SQBM+ computation API you are using. ↩
4. ↩
5. Substitute private_key.pem with the file path to the private key for connecting to the SQBM+ server; substitute sqbmplus_server with the host name of the SQBM+ server. ↩