SQBM+ for On-premises - ユーザーガイド

Note

目次

はじめに

SQBM+は大規模組み合わせ最適化問題に対するソルバー一式を提供します。SQBM+のソルバーにより、ユーザーは最適解もしくは良解を得ることができます。 SQBM+は以下の挙げる論文で示されている理論に基づいて開発されています。

- 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+はAmazon Elastic Compute Cloud (Amazon EC2)で使用されるAmazon Machine Image(AMI)もしくはRPMパッケージとして提供されます。SQBM+はFigure 1の構成になっています。

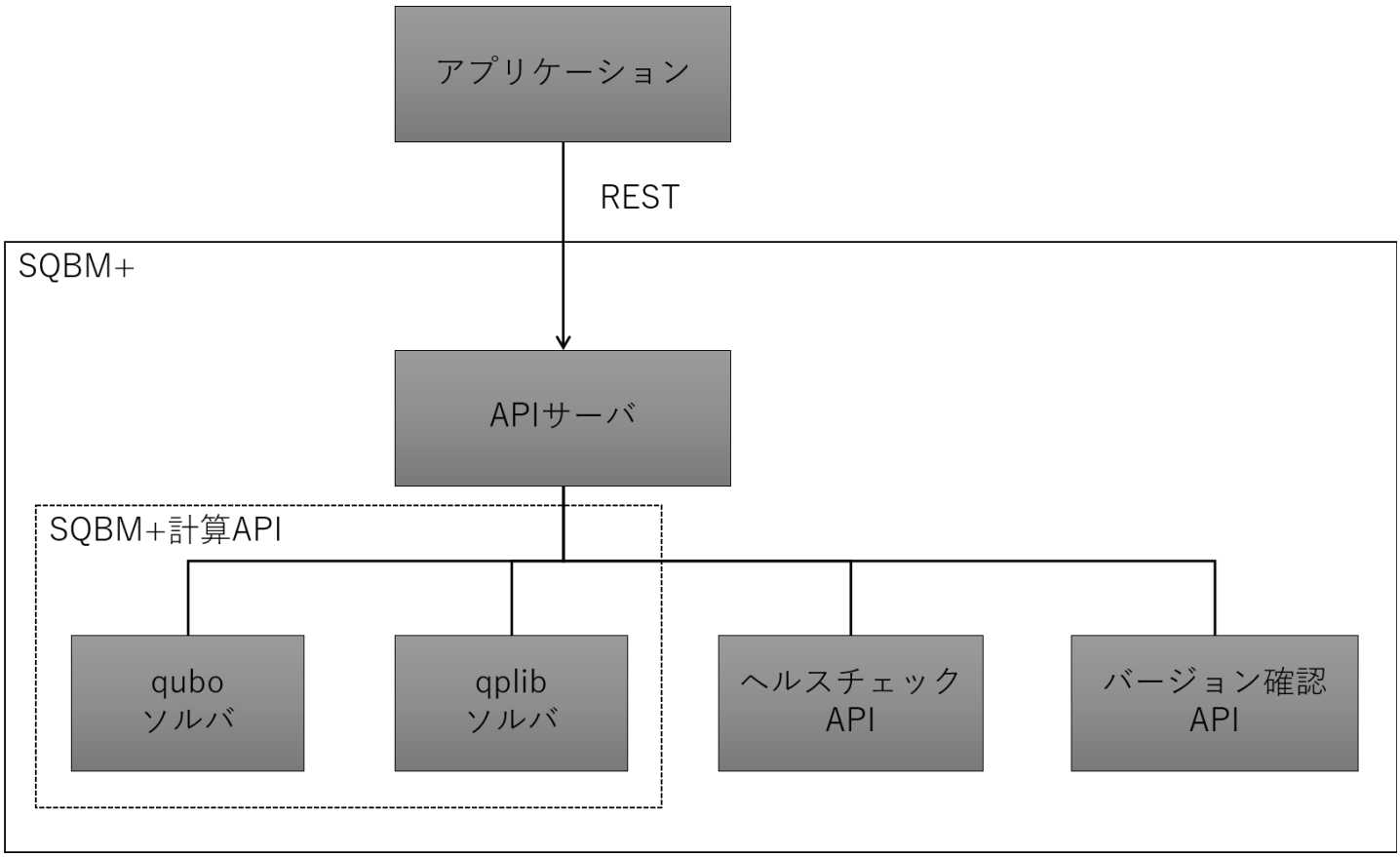

Figure 1 SQBM+の機能ブロックダイアグラム

SQBM+は主に以下の特徴があります。

- RESTful APIとしてSQBM+計算APIを提供しています。
- 様々な最適化問題とデータフォーマットに対応したソルバーとインターフェースを備えています。現在は以下のソルバーを提供中です。詳細は各ソルバーのセクションを参照してください。
  - quboソルバー: quboモデルに対応したソルバー
  - qplibソルバー: 2次計画問題に対応したソルバー
- 大規模最適化問題の最適解もしくは良解をすばやく得ることができます。[1]
- 簡単に使用でき、複雑なパラメーター設定は不要です。

本ドキュメントは以下の構成となっています。

- SQBM+概要
- SQBM+計算API
- ヘルスチェックAPI
- バージョン確認API
- システム構成パラメーター

はじめに「SQBM+概要」でSQBM+が提供するAPIの概要を紹介します。ここでユーザーはSQBM+が提供するAPI利用方法とその応答について概要を把握することができます。続けてSQBM+計算API、ヘルスチェックAPI、バージョン確認APIの３つAPIについて説明致します。ユーザーは各APIを説明している章まで読むことにより、詳しい利用方法を把握することができます。

最後に「システム構成パラメーター」でSQBM+のシステム設定について紹介します。ユーザーはシステム設定をすることなくSQBM+を使用することができます。しかしながら、より大規模な問題を扱う場合や計算時間の制限を変えたい場合など、デフォルトの設定から変更する必要がある場合があります。この章を確認し、設定変更の要否をご確認ください。

SQBM+計算APIとの接続に関する注意

SQBM+は認証を行わないので、インターネットから直接接続することは推奨していません。
例えばAWS環境の場合、SQBM+のEC2インスタンスが設置されているVPC内のインスタンスから接続することを推奨します。

SQBM+概要

SQBM+は以下の3つのAPIを提供します。

- SQBM+計算API: 組み合わせ最適化問題の解を求めるAPI
- ヘルスチェックAPI: SQBM+計算APIの稼働状況を確認するAPI
- バージョン確認API: ご使用中のSQBM+のバージョンを確認するAPI

いずれのAPIもクライアントからSQBM+サーバーにHTTPリクエストを送信することで利用することができます。
各APIからの応答はHTTPレスポンスとして返されます。
Figure 2はSQBM+計算APIの利用例となります。APIを利用するクライアントがSQBM+サーバーに対しHTTPリクエストを送信しています。この例では最適化問題を解くようにリクエストを送信しています。HTTPリクエストを受け取ったSQBM+サーバーは計算処理を実施し、求めた解をクライアントに返します。応答はJSONフォーマットとなります。

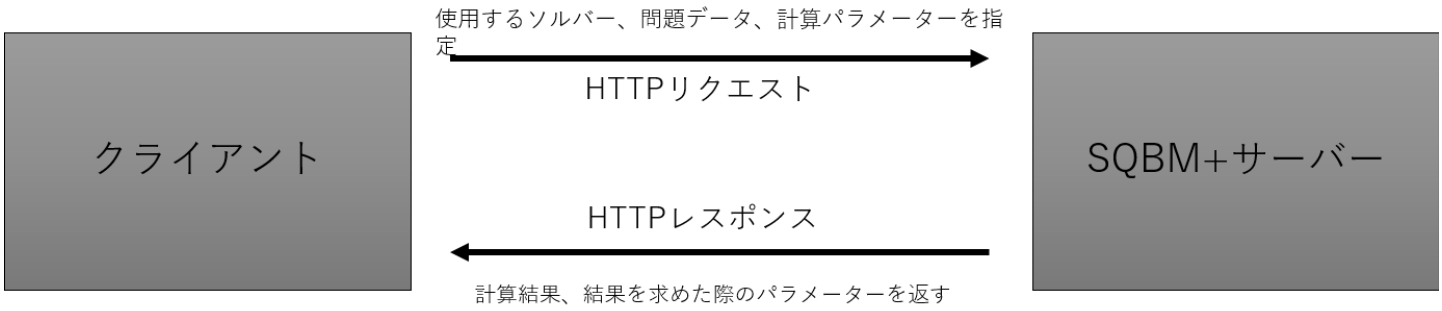

Figure 2 SQBM+計算APIの利用例

SQBM+が提供しているAPIをまとめたものがTable 1となります。
各APIにおける詳細はSQBM+計算API、ヘルスチェックAPI、バージョン確認APIの章をご確認ください。

Table 1 SQBM+ API情報

API種別	HTTPメソッド	URL path	Port	character code
SQBM+計算API	POST	http://{サーバーのIPアドレス}:{ポート番号}/solver/{solver} {solver}: 使用するソルバーとしてqubo、qplibのどちらかを選択してください。 {ポート番号}: デフォルト8000となります。	8000 (default)	UTF-8
ヘルスチェックAPI	GET	http://{サーバーのIPアドレス}:{ポート番号}/healthcheck {ポート番号}: デフォルト8000となります。	8000 (default)	UTF-8
バージョン確認API	GET	http://{サーバーのIPアドレス}:{ポート番号}/version {ポート番号}: デフォルト8000となります。	8000 (default)	UTF-8

SQBM+の各APIは以下の特徴を持ちます。

- SQBM+はセッション情報を持ちません。
- SQBM+計算API実行時のクエリパラメーター情報、問題データについての統計情報、実行状況についての情報は時刻付きでログに残りますが、問題データそのものは保存しません。またユーザーへの応答データも保存しません。過去のリクエスト結果を問い合わせることはできず、データも残りませんのでご注意ください。
- HTTPリクエストを送信することによりSQBM+の各APIを利用します。各APIへのアクセスはTable 1のURL pathををご利用ください。

SQBM+計算APIでは解きたい問題に合わせて、ソルバーを選択してください。
SQBM+計算APIは計算制御に関わるパラメーターをHTTPリクエストのクエリ文字列で設定することができます。
SQBM+計算APIに解かせたい問題データはHTTPリクエストボディとして設定します。
SQBM+計算APIはFigure 3のように全ソルバー共通の仕様が定義され、それに加え各ソルバー固有の仕様が定義されています。レスポンスに関する仕様、使用できるパラメーターは全ソルバーに共通したものと各ソルバー固有のものがあります。また、リクエストボディとして設定できる問題データのフォーマットは各ソルバーで定められています。ユーザーは全ソルバー共通仕様を確認し、その上で利用したいquboソルバー、qplibソルバー固有の仕様を把握する必要があります。

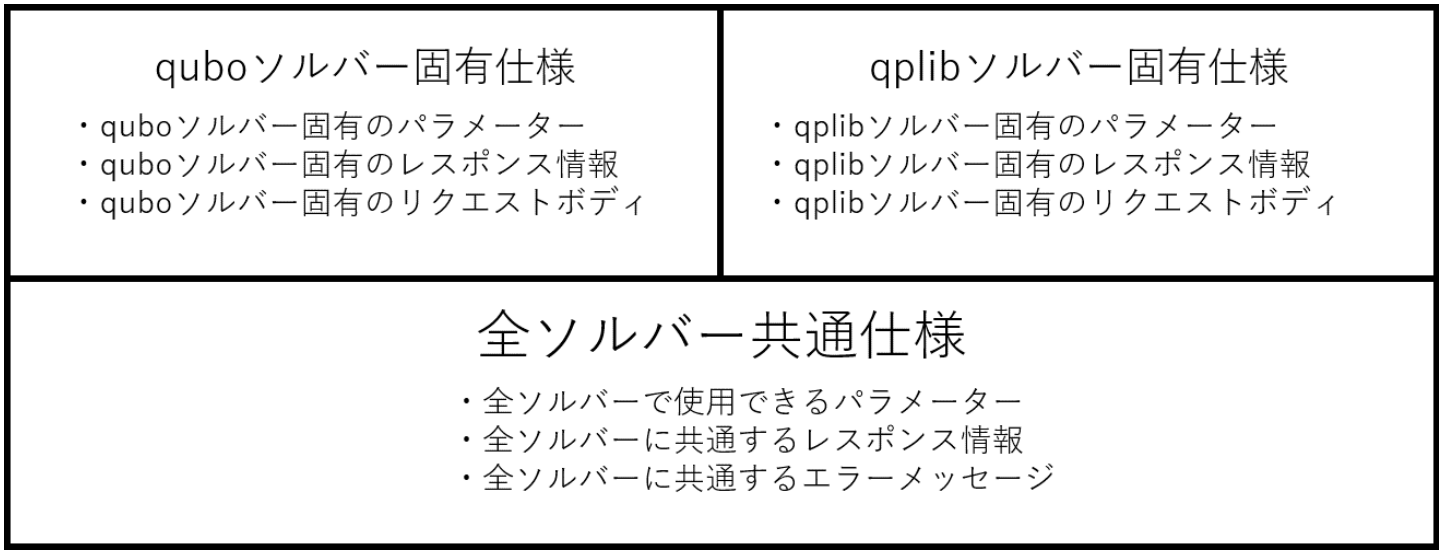

Figure 3 SQBM+計算APIの仕様階層

SQBM+計算API

SQBM+はメインAPIとして組み合わせ最適化問題を解くソルバーを提供します。quboソルバー、qplibソルバーの各章を確認し、解きたい問題に合ったソルバーをご使用ください。各ソルバーは様々なフォーマットの問題データを受け付けます。各ソルバーは自動で問題データのフォーマットを認識しますので、ユーザーはHTTPリクエスト時にデータフォーマット指定をすることはありません。使用できるフォーマットについてもquboソルバー、qplibソルバーの各章を確認してください。

SQBM+による求解に必要な情報は以下のようになっています。

- 使用するソルバーと問題データ(必須)
- SQBM+による求解を制御するパラメーター指定(オプション)

ユーザーは全ソルバー共通仕様の章を確認し、全ソルバーに共通したHTTPリクエスト、レスポンスの仕様を把握する必要があります。その上で利用したいquboソルバー、qplibソルバー固有の仕様を確認することで、各ソルバーに固有な制御パラメーターの情報とレスポンス詳細を把握できます。
求解を制御するパラメーターの指定がない場合は、システム構成パラメーターの設定値が適用されるか、もしくはSQBM+内部で自動調節されます。そのため、パラメーター指定は必須ではありません。SQBM+計算APIは、レスポンスに解を求めた際のパラメーター一覧を含めます。ユーザーがパラメーター指定する場合には、レスポンスに含まれるパラメーター情報を参考にすることができます。
SQBM+計算APIの制御に大きく関わるパラメーターとして計算シーケンス制御に関わるものがあります。これはSQBM+計算APIの解探索方法の大枠を決めるものです。SQBM+計算APIの計算構造で、このパラメーターについて説明します。

SQBM+計算APIの計算構造

SQBM+計算APIは2つのスコープで計算シーケンスを制御することができます。1つは求める解の数を制御します。もう1つは繰り返し実行するSBアルゴリズム処理の回数を制御します。SBアルゴリズム処理1回で決定変数が 1 回更新されます。

- SQBM+計算APIは求める解の数を指定できます。
  指定は後述するloopsパラメーター経由で実施します。SQBM+計算APIが求める解の数はお使いの環境のGPUデバイス数とGPU内のマルチプロセッサー数に依存します。少なくともloops × GPUデバイス数 × マルチプロセッサー数以上の解を得るまで繰り返し最適解の探索を実施します。
  デフォルト動作ではSQBM+計算APIは求めた解の中から最良の解1個を選んでユーザーに返しますが、評価値の良いものから順に指定された数の解を返すことも可能です。詳細はシステム構成パラメーター、リクエストパラメーターを参照ください。
- SQBM+計算APIは繰り返し実行するSBアルゴリズム処理の回数を指定できます。
  指定は後述するstepsパラメーター経由で実施します。stepsが多ければ多いほど1つの最適解探索に時間をかけることになります。

stepsとloopsは求める解の精度と計算時間に影響します。SQBM+計算APIはパラメーターの自動設定機能を備えており、よりよいパラメーターの設定を見つける助けになります。詳細はリクエストパラメーターを参照ください。
SQBM+計算APIが最終的に求めた解の数はrunsというパラメーター値で表現されます。1つの求解を1runとカウントしています。これはSQBM+計算APIのレスポンスとして返されます。詳しくはレスポンス仕様を参照ください。SQBM+計算API構造は Figure 4のようになります。

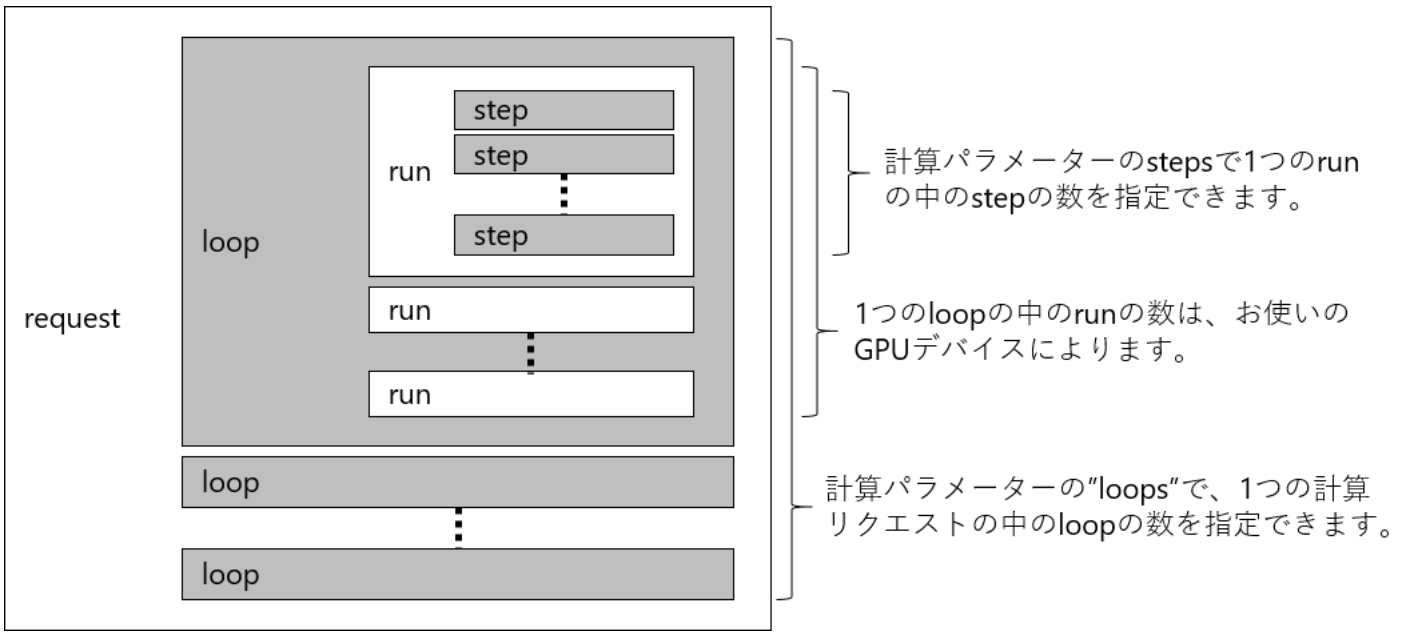

Figure 4 SQBM+計算APIの計算構造

全ソルバー共通仕様

SQBM+計算APIへのリクエストはRESTful APIを使用して行います。
RESTful APIとして使用するHTTPメソッドとURL構成、全ソルバー共通パラメーターはリクエスト仕様のようになっています。

リクエスト仕様

HTTPリクエスト

- 使用するHTTP メソッド: POST
- URL構成: http://{ip}:{port}/solver/{solver}?{parameter}
  - ip: SQBM+計算APIが稼働しているマシンのIPアドレス
  - port: SQBM+サーバーのポート番号
  - solver: quboもしくはqplib
  - parameter: key=valueのペアで構成されるクエリ文字列。使用できるパラメーターの種類と詳細はリクエストパラメーター及び各ソルバー固有仕様を参照。値を指定しなかった場合はデフォルト設定値が適用されます。

リクエストパラメーター

Table 2 全ソルバー共通パラメーター

Name	Default	Description
steps	0	SQBM+計算APIの計算構造のstepsを指定します。0は自動ステップ指定を意味し、 SQBM+計算APIが解を求める度に自動的にstep数を変化させて計算します。 0以上、100,000,000以下の整数を指定してください。
loops	0	SQBM+計算APIの計算構造のloopsを指定します。もし0が指定された場合は、loops＝ 100,000,000となります。 0以上、100,000,000以下の整数を指定してください。
timeout	10秒 デフォルト値は システム設定ファイルで変更可能	最大計算時間（タイムアウト）を秒で指定します。 loopsで指定された計算を終える前に最大計算時間に達した場合、そこで計算を終了します。 この場合、それまでに得られた最良の解が実行結果となります。 このtimeoutとは別にシステムの計算時間の上限値があります。 詳細についてはシステム構成パラメーターの timeout_upper_boundをご確認ください。 timeoutがtimeout_upper_bound以上の場合、 timeout_upper_boundがtimeoutとして採用されます。 設定時間が長い場合、SQBM+が求めることができる最大求解数に到達し、 timeout時間未満で計算を終了することがあります。 この場合もそれまでに得られた最良の解が実行結果となります。 0より大きい値を指定してください。
maxwait	60秒 デフォルト値は システム設定ファイルで変更可能	各計算リクエストに対する最大待ち時間を秒で指定します。 すでに実行中である計算リクエストがある場合に、新たに計算リクエストを投入すると、 そのリクエストは実行待ち状態になります。 この待ち時間がmaxwaitを超えた場合はエラー終了します。 0より大きく100,000,000以下を指定してください。
target	設定なし	targetを指定した場合、目的関数の評価値がtargetに到達したら計算を終了します。 loopsが0である場合、 目的関数の評価値がtargetで指定した値に達するかタイムアウトするまで求解を繰り返します。
maxout	1	出力する解の上限数を指定します。maxoutで指定された数を上限として、 見つけた解のうち目的関数の評価値の良いものから順に解を出力します。 1以上の整数値を指定してください。

問題データ指定

組み合わせ最適化問題データは各ソルバーでサポートしているデータフォーマットで作成してください。
ただし、指定したデータは32bit、もしくは64bitで表現される範囲の小数に丸め込まれます。
組み合わせ最適化問題データの設定は以下の２つの方法で実施できます。

- リクエストボディでの設定
- URIでの問題ファイルパス指定

リクエストボディでの設定

リクエストボディで問題データを渡す場合は、問題データを設定してください。

URIでの問題ファイルパス指定

組み合わせ最適化問題データファイルをSQBM＋サーバーがアクセスできる場所に置き、ファイルパスをHTTPヘッダーとして設定し、問題データをSQBM＋に読み込ませることができます。
ヘッダーとしてTable 3のヘッダー情報を設定してください。

Table 3 設定対象リクエストヘッダーと設定値

Header field	Value
X-Input-Data-URI	絶対ファイルパス

レスポンス仕様

全ソルバー共通レスポンスステータス

計算が正常終了した場合、HTTPステータスコード200が返されます。
計算が異常終了した場合は200以外のHTTPレスポンスステータスコードが返されます。
詳細はTable 6をご確認ください。

全ソルバー共通レスポンスヘッダー

計算正常終了時にはTable 4のヘッダーが設定されます。

Table 4 設定対象レスポンスヘッダーと設定値

Header field	Value
Content-Type	application/json
X-Calculation-Time	実計算時間。レスポンスボディのtimeと同じ
X-ID	内部的に割り当てられたリクエストID

計算異常時にはContent-Typeの値として"text/html; charset=utf-8"が設定され、X-Calculation-Timeヘッダー、X-IDヘッダーはありません。

全ソルバー共通レスポンスボディ

Table 5は計算が正常終了した際にレスポンスボディに格納されるプロパティです。結果はJSON形式で返されます。Table 5で示されているプロパティは全ソルバーが共通して持ちます。

Table 5 全ソルバー共通計算結果プロパティ

Property name	Description
id	内部的に割り当てられたリクエストID
time	データ転送時間などのオーバヘッドを含まない実計算時間（秒）
wait	SQBM+サーバーが問題をロードしてから、SQBM+計算が開始されるまでの処理待ち時間（秒）
message	SQBM+計算APIの終了状態を表すメッセージ。finished、timeout、reached、max results、 saturatedの5種類のメッセージのうち1つが設定されます。各メッセージの意味は以下のようになっています。: - "finished": 指定された数のstepsとloopsの計算を完了した。 - "timeout": タイムアウト時間で計算を終了した。 - "reached": 評価値valueがパラメーターtargetで示す値に到達したので計算を終了した。 - "max results"：1つのリクエストで求められる最大個数の解を得た。 - "saturated": これ以上良い解が得られる見込みがなく、計算が収束した。
runs	求めた解の数
value	本表中のresultで表される解に基づいて計算された評価値
result	得られた最良の解
param	最良の解を得た際に使用した各パラメーター設定値。JSON形式。 表示されるパラメーターは使用するソルバーとアルゴリズムによって異なります。

Property name	Description
count	最良の解を得た回数
others	その他の解の評価値と変数の組、param、その解を得た回数のリスト。 ここに含まれる解の個数の合計はリクエストパラメーターmaxout-1以下になります。例としては複数解出力例を参照ください。

SQBM+計算APIへの入力データとSQBM+計算APIからの出力データはSQBM+に保存されないので、ご注意ください。

エラーメッセージ

SQBM+計算APIに異常があった場合はHTTPのレスポンスステータスコードとして200以外が返されます。HTTPのレスポンスボディにはエラーの原因を示すメッセージが入ります。Table 6にエラーメッセージ一覧を示しています。

Table 6 エラーメッセージ一覧

Status code	Message	Description
400	入力パラメーター、 入力問題データに関してのエラーメッセージ	入力パラメーター、入力問題データが正しくありません。 メッセージに詳細な情報が入ります。
400	no result	解を得ることができませんでした。timeoutが短い、 もしくは線形制約が厳しい条件になっている可能性があります。
403	The evaluation period has passed.	試用期限を過ぎています。評価版でのみ発生します
404	URL異常に関するメッセージ	リクエストされたURLが不正です。
413	payload too large	問題ファイルのサイズがpayload_limitを超えています。 詳細はシステム構成パラメーターをご確認ください。
500	サーバー内部処理エラーメッセージ	このエラーが発生した場合はSQBM+サーバーが予期していないエラーが発生しています。 サーバー管理者にご相談ください。
503	busy	同時に受付できるリクエスト数を超えています。
503	engine stopped	エンジンが停止しました。この場合SQBM+は自動で再起動を試みます。 このエラーに関する詳細はSQBM+計算APIが稼働しているサーバー内のsbm-engine.logをご確認ください。詳細はログファイルをご確認ください。
504	timeout	maxwaitの待ち時間が経過し、リクエストに対するSQBM+計算は終了となりました。

quboソルバー

quboソルバーはquboモデルで表現された組み合わせ最適化問題を解きます。

quboソルバーのリクエスト仕様

HTTPリクエスト

http://{ip}:{port}/solver/{solver}?{parameter}
において、{solver}としてquboを選択してください。

リクエストヘッダー

Table 7 設定対象リクエストヘッダーと設定値

Header field	Value
Content-Type	application/octet-stream

リクエストパラメーター

全ソルバー共通パラメーター(Table 3)に加え、Table 8に示しているソルバー固有パラメーターがquboソルバーでは使用できます。

Table 8 ソルバー固有パラメーター

Parameter	Default	Description
dt	0	SBアルゴリズムの時間発展に使用されるタイムステップ幅を指定します。 0はタイムステップ幅の自動調節指定を意味し、SQBM+計算APIが内部で自動的にdtを変化させて計算します。 0.0以上1.5以下の値を指定してください。
C	0	SQBM+のベース理論であるGoto, Tatsumura, & Dixon (2019)の論文で使用されているξ_0に対応します。 0が指定された場合、Cの値は自動調整されます。 0以上の単精度浮動小数点数を設定してください。
algo	0	Goto et al. (2021)の論文で発表されているSQBM+の計算アルゴリズム15 (bSB)もしくは20 (dSB) をはじめとしたSQBM+内部で実装されているアルゴリズムを指定します。指定できるアルゴリズムはTable 9をご確認ください。使用するアルゴリズムによって計算結果が変わる場合があります。0が指定された場合、SQBM+は様々なアルゴリズムを使用して解探索を実施します。

algos	-	正規表現を用いて計算アルゴリズムを複数指定できます。このパラメーターを指定した場合algoではなく、algosで指定した複数アルゴリズムを使用します。
		`http://localhost:8000/solver/qplib?timeout=1&algos=1.`
		を
		`http://localhost:8000/solver/qplib?timeout=1&algos=1%2e`
		のようにURLエンコードして指定してください。

Table 9 使用できるアルゴリズム

Algorithm	Description
15	Goto et al. (2021)の論文で発表されているSQBM+の計算アルゴリズムbSB
151	algo=15の処理に加え、CをSBアルゴリズム更新ステップ毎に自動調節します。これにより、 固定のCを指定するよりも精度が改善することがあります。
154	algo=15の処理に加え、SBアルゴリズムでのQUBO係数行列の影響度を自動的にスケーリングします。 QUBO係数行列の固有値の分布に偏りがある場合に、精度が改善することがあります。
155	151と154の両方を実施します。
20	Goto et al. (2021)の論文で発表されているSQBM+の計算アルゴリズムdSB
201	algo=20の処理に加え、CをSBアルゴリズム更新ステップ毎に自動調節します。これにより、 固定のCを指定するよりも精度が改善することがあります。
204	algo=20の処理に加え、SBアルゴリズムでのQUBO係数行列の影響度を自動的にスケーリングします。 QUBO係数行列の固有値の分布に偏りがある場合に、精度が改善することがあります。

Algorithm	Description
205	201と204の両方を実施します。

問題データ指定

問題データ指定では、問題データに当たる係数行列をQUBO形式で表現する必要があります。与えられた問題に対するエネルギー計算はQUBO行列を使用して計算されます。最小化の対象となるエネルギーは以下の式となります。:

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

Q_{ij}は行列の要素、$x_i \in \{0, 1\}$はqubo変数を表します。 この式は以下の式に変換できます。:

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

この式はQUBO行列の対角成分を分離して、線形項を表記した形となります。
quboソルバーはMatrixMarketとHDF5の２つのデータフォーマットをサポートしています。
QUBO行列の各要素は絶対値が1e+20以下である必要があります。

MatrixMarketフォーマット

問題データはMatrixMarketフォーマットを使用することができます。詳細についてはhttps://math.nist.gov/MatrixMarket/formats.html をご確認ください。

- SQBM+はsymmetryとしてgeneralのみサポートしています。ただしquboソルバーが返す評価値であるvalueは対角成分を含む下三角行列の要素のみを使用して計算されます。symmetric指定は後方互換対応としてサポートしていますが、今後廃止予定です。なおsymmetric指定した場合でもgeneralと同様に処理されます。
- 実問題をMatrixMarketフォーマットへ変換する方法については、quboソルバーの使用例に記載されているMAX-CUT問題の例を参照してください。
- データタイプにintegerが指定されているが実際のデータがrealの場合は、quboソルバーはエラーを返さずに実際のデータタイプを優先してrealとして計算を行います。
- 行列の要素をインデックスで指定するとき、同じ要素を重複して指定しないように注意してください。重複があった場合のquboソルバーの動作は不定で、ステータスコード400のエラーが返る場合があります。

quboソルバー対応HDF5フォーマット

HDF5フォーマットのバイナリ問題データをサポートしています[2]。フォーマットの詳細については https://www.hdfgroup.org/solutions/hdf5/ をご確認ください。

- SQBM+は入力ファイルのヘッダーを確認し、入力データフォーマットを認識します。そのため、ユーザーはフォーマットタイプを指定する必要はありません。
- SQBM+が受けつけるHDF5ファイルは以下の2つの構成のどちらかでなければなりません。

疎QUBO行列に対応したファイル構成

疎行列の表現にはCSRフォーマットを使用しています。詳細は
https://en.wikipedia.org/wiki/Sparse_matrix#Compressed_sparse_row_(CSR,_CRS_or_Yale_format) をご確認ください。参照先のROW_INDEXが/qubo/indptr、COL_INDEXが/qubo/indices、Vが/qubo/dataに対応しています。ファイル構成は以下のようになっています。

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

/qubo/indptr、/qubo/indices、/qubo/dataは１次元配列です。

密QUBO行列に対応したファイル構成

ファイル構成は以下のようになっています。

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

/qubo/dataはQUBO行列に対応した2次元配列です。

データタイプ

```
疎QUBO行列に対応したファイル構成を使用する際には各配列の要素は以下のデータタイプを使用してください。
data array: float32
indices array: uint32
indptr array: uint32
密QUBO行列に対応したファイル構成を使用する際には各配列の要素は以下のデータタイプを使用してください。
data array: float32
```

実際の使用方法については HDF5 fileを入力データとして使用した例 をご確認ください。

入力問題データに関する制限

quboソルバーが扱える問題サイズには以下の制限があります。

- qubo変数の数は100,000以下である必要があります。
- QUBO行列の非ゼロ要素数は800,000,000以下である必要があります。
  ただしメモリの関係上800,000,000以下の場合でも解が得られない場合があります。

quboソルバーのレスポンス仕様

レスポンスステータス

全ソルバー共通レスポンスステータスと同様です。

レスポンスヘッダー

全ソルバー共通レスポンスヘッダーと同様です。

レスポンスボディ

Table 10の項目を除き、全ソルバー共通レスポンスボディと同様です。

Table 10 quboソルバーのレスポンスボディ仕様

Property name	Description
value	QUBOモデルで計算された評価値になります。
result	各qubo変数の配列。qubo変数が5つのときには、resultは `[0,1,1,0,0]` にようになります。
param	stepsとquboソルバー固有パラメーターとしてalgo、dt、Cが表示されます。

quboソルバーの使用例

MAX-CUT問題を例としてquboソルバーの使用方法を説明します。MAX-CUT問題とはグラフの頂点を2つのグループに分割するときに、カットされる辺の数が最大になるような切り方を探す問題です。重み付MAX-CUT問題は辺に重みがついており、カットされる辺の重みの総和が最大になる切り方を探す問題になります。Figure 5 (すべての重みが1の場合)のMAX-CUT問題は最適解が5つあります。

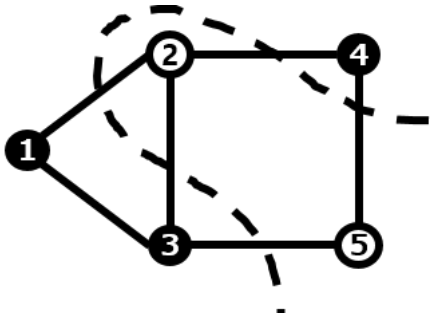

Figure 5 MAX-CUT問題

MAX-CUT問題の目的関数はIsingモデルにより以下のように表現できます。

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

$s_i \in \{-1, 1\}$はスピン変数であり、その値は頂点iが2等分されるグループのどちらに所属するかを示します。W_{ij}は辺ijの重みを示します。もし頂点iと頂点jが隣接しており、かつ辺で繋がっていれば、同じグループに所属することになります。その場合$s_i s_j = 1$となります。もしiとjが異なるグループに所属している場合は、$s_i s_j = -1$となります。これは、隣接している頂点が別のグループに属している程、つまり、より多くの2頂点間の辺が切られている程、エネルギーEは小さくなることを示しています。
SQBM+のquboソルバーへの入力問題はQUBO形式である必要があります。そのためIsingモデルはQUBOモデルに変換される必要があります。QUBO

形式では変数$x_i \in \{0,1\}$であるので、スピン変数は$s_i = 2x_i - 1$と表現できます。これをIsingモデルに代入し式を整理すると、以下のQUBO形式の目的関数が得られます。

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

上式からquboソルバーへの入力であるQUBO行列を得ることができます。Figure 5の例の場合、QUBO行列はFigure 6のようになります。1次項の係数は対角成分を構成します。重みW_{ij}はiとjが辺で繋がっていれば1としています。

i \ j					
	-2				
	2	-3			
	2	2	-3		
		2		-2	
			2	2	-2

Figure 6 QUBO行列

Figure 7はFigure 6のQUBO行列から作成したのquboソルバーへの入力ファイルを示しています。qubo.txtがファイル名であり、MatrixMarketフォーマットで作成しています。

```
$ cat 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
```

Figure 7 Matrix Marketフォーマットの例

Figure 7のqubo.txtを使用し、quboソルバーでMAX-CUT problem[3]を解いた結果は以下のようになります。

> $ 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}

このように[1,1,0,0,1]の実行解を得ることができます。結果はFigure 5の最適化結果を示しています。このサンプル問題では複数の最適解が存在するので、いつでも同じ結果が得られるわけではありません。

パラメーターの設定例

ここから先は、quboソルバーのパラメーター設定例を紹介します。

stepsとloopsの設定例

```
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/"b8-5IW9ZBtd9mXZhimG1zlj0xEQTDA"
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}
```

CとdtのSetting例

```
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/lE7hI/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}
```

リピート計算例

targetで指定した評価値(value)が得られるかタイムアウトになるまで計算を続けます。

```
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}
```

複数解出力例

```
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}]}

HDF5 fileを入力データとして使用した例

> 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}

MatrixMarketフォーマットファイルからHDF5 fileへの変換には以下のサンプルプログラムを使用しています。

```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):
    '''
    行列を下三角行列に変換する。ただしxMxの値は変わらないようにする。

    Argument:
        M (Numpy array): 行列

    Return:
        MM (Numpy array): 下三角行列
    '''
    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):
    """
    MatrixMarketフォーマットを想定

    Argument:
        file: MatrixMarketフォーマット
    Return:
        H (Numpy array): 目的関数の2次項を表す行列
    """
    stm = (line.strip() for line in file) #ジェネレーター

    # %%MatrixMarket matrix coordinate real symmetricの分はいらない
    _ = 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):
    '''
    SBMに入力するHDF5ファイルを作成する
    このメソッドではファイルへの行列データの書き込みを実施する。

    Argument:
        M (Numpy array): QUBO行列
        quboFileName (String): 出力先のhdf5形式ファイル名
        path (String): hdf5のグループ、サブグループで構成するパス。例 "category1/category2"
        comp_flag (Boolean): 圧縮をするかどうかを定める。 True or False
        form (String): 行列を保持する形式指定。CSR形式か、0を含め全要素を保持（dense)するか
    Return:

    '''
    if form == 'dense':
        with h5py.File(quboFileName, mode='a') as hf:
            group = hf.create_group(f'/{path}')
            # dataの型は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)
        # 以下のレイアウトで保存すること
        with h5py.File(quboFileName, mode='a') as hf:
            group = hf.create_group(f'/{path}')
            # dataの型は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の型は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の型は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

    # 出力ファイル名を作成
    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)
            # 制約を一つのディレクトリにまとめる
            write_matrix_HDF5(A_o, REQ_HDF5, '/constraints/coeff', comp, form)#係数
            write_vector_HDF5(cl_o, REQ_HDF5, '/constraints/lower', comp)#下限
            write_vector_HDF5(cu_o, REQ_HDF5, '/constraints/upper', comp)#上限

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

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

qplibソルバー

qplibソルバーはバイナリ変数、2次関数の目的関数と線形制約から構成される2次計画問題を解きます。qplibソルバーは目的関数のQUBO係数行列とは別に線形制約を受け取り、発見できた最良解を返します。

問題の定義 (線形制約付きバイナリ2次計画問題)

- 変数:

$$x \in \{0,1\}^N$$

- パラメーター:

$$Q, A, LHS, RHS$$

- 制約付き2次計画問題:

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

目的関数に使用されているQはQUBO行列、Bは線形項の係数ベクトルです。
線形制約条件に使用されているAは$M \times N$行列です。Mは制約数であり、Nは変数の数です。この行列により

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

のような複数の線形制約をまとめて $LHS \leq A \cdot x \leq RHS$ と表現しています。
上記の例では $LHS = \{-1, -3, -4\}$、$RHS = \{1, 1, 1\}$のベクトルです。

- ここで、LHS, RHS は以下のどれかを満たしている必要があります。ここでは LHS ベクトルのm番目成分を LHS_m というように表現しています。
  - 上限、下限が定義されているときの不等式制約
    $LHS_m < RHS_m,\ LHS_m \in \mathbb{R},\ RHS_m \in \mathbb{R}$
  - 上限が定義されていないときの不等式制約
    $LHS_m \in \mathbb{R}, RHS_m = +\inf$
  - 下限が定義されていないときの不等式制約
    $LHS_m = -\inf, RHS_m \in \mathbb{R}$
  - 等式制約
    $LHS_m = RHS_m,\ LHS_m \in \mathbb{R},\ RHS_m \in \mathbb{R}$

QUBO行列、Bベクトルの各要素は絶対値が1e+20以下である必要があります。
A行列、LHS、RHSベクトルの各要素は絶対値が1e+20を超えている場合はA行列、LHS、RHSの各要素が1e+20を超えないように制約条件を保ちつつスケーリングします。

qplibソルバーのリクエスト仕様

HTTPリクエスト

http://{ip}:{port}/solver/{solver}?{parameter}
において、{solver}としてqplibを選択してください。

リクエストヘッダー

Table 11 設定対象リクエストヘッダーと設定値

Header field	Value
Content-Type	application/octet-stream

リクエストパラメーター

quboソルバーと同様です。

問題データ指定

問題データ指定はHDF5フォーマットファイルとqplibフォーマットファイルを使用することができます。

HDF5フォーマット

qplibソルバーが使用できるHDF5ファイルは以下の構成になっている必要があります。
infを指定する場合は3.40283e38を指定してください。

Table 12 qplibソルバーが使用するHDF5ファイルフォーマット

Structure	Group name or the content
/qubo	Q 行列データ
/linear	B ベクトルデータ
/constraints	制約条件のグループ名
/coeff	A 行列データ
/lower	LHS ベクトルデータ
/upper	RHS ベクトルデータ

HDF5ファイルの各データセットは以下の規則に従う必要があります。

Table 13 データタイプ毎のアトリビュート設定

Data	Data type	Content attributes
ベクトル	array	ベクトルを1次元配列として定義してください。各成分のデータタイプはfloat32です。 アトリビュートは "format"に"dense" を設定してください。
疎行列	Sparse array	quboソルバー対応HDF5フォーマットと同様。 アトリビュートは "format"に"csr" を設定してください。
密行列	Dense array	quboソルバー対応HDF5フォーマットと同様。 アトリビュートは "format"に"dense" を設定してください。
Q 行列データ	Dense array Sparse array	密行列としてデータを設定する場合はDense arrayデータタイプとしてデータを設定してください。 疎行列としてデータを設定する場合はSparse arrayデータタイプとしてデータを設定してください。
B ベクトル	array	"format"に"dense" を設定してください。
A 行列データ	Dense array Sparse array	密行列としてデータを設定する場合はDense arrayデータタイプとしてデータを設定してください。 疎行列としてデータを設定する場合はSparse arrayデータタイプとしてデータを設定してください。
LHS ベクトル	array	"format"に"dense" を設定してください。
RHS ベクトル	array	"format"に"dense" を設定してください。

qplibフォーマット

qplibソルバーはQPLIB[4]が使用するqplibフォーマットに対応しています。詳細は http://qplib.zib.de/doc.html をご確認ください。qplibフォーマットは目的関数や変数の型、制約のタイプで問題タイプを分けています。問題タイプはObjective typeとVariables typeとConstraints typeの３つを指定します。各指定はアルファベット１文字で指定します。指定できるタイプと指定文字の詳細は http://qplib.zib.de/doc.html をご確認ください。現在、qplibソルバーは['QBB', 'QBL']の問題タイプが指定できます。qplibフォーマットファイルでは、ファイルの先頭に以下の3つの設定項目を記入してください。

- 問題名: "QPLIB_"から始まる文字列
- 問題タイプ指定: ['QBB', 'QBL']から選択してください。
- 目的関数の最小値を求めたい場合はminimize、最大値を求めたい場合はmaximizeを指定します。

その後、Table 14のタグに従って問題設定をファイルに記載してください。
infを指定する場合は1.79769313486232E+308を指定してください。

Table 14　設定タグ情報

tag	Description
number of variables	使用する変数の数を指定してください。 2以上100,000以下の整数を指定してください。 例 3 # number of variables

tag	Description
number of constraints	使用する線形制約条件の数を指定してください。但しConstraints typeがB(線形制約条件なし)である場合、 0以外の指定するとエラーとなります。 0以上1,000,000以下の整数を指定してください。 例 3 # number of constraints
default left-hand-side value	全線形制約条件の下限値設定をします。LHSベクトル全成分のデフォルト設定値を指定してください。但しConstraints typeがB(線形制約条件なし)である場合はこのタグを使用しないでください。 実数を指定してください。 Constraints type 例 0 # default left-hand-side value
default right-hand-side value	全線形制約条件の上限値設定をします。RHSベクトル全成分のデフォルト設定値を指定してください。但しConstraints typeがB(線形制約条件なし)である場合はこのタグを使用しないでください。 実数を指定してください。 例 0 # default right-hand-side value
default value for linear coefficients in objective	Bベクトル全成分のデフォルト設定値を指定してください。 絶対値が1E+20以下の実数を指定してください。 例 0 # default value for linear coefficients in objective
default variable type	各変数のデフォルトデータ型を指定してください。但しVariables typeがB(バイナリ)、C(連続変数) である場合はこのタグを使用しないでください。 以下から選択できます。 0: 連続実数 1: バイナリ 例 1 # default variable type
default variable upper bound value	全変数の定義域上限値を設定します。但しVariables typeがB(バイナリ)である場合は、このタグを使用しないでください。 -1E+20以上の実数を指定してください。 例 1.5 # default variable upper bound value
default variable lower bound value	全変数の定義域下限値を設定します。但しVariables typeがB(バイナリ)である場合は、このタグを使用しないでください。 1E+20以下の実数を指定してください。 例 1.5 # default variable lower bound value
number of quadratic terms in objective	QUBO行列を設定します。各行列成分の設定値を指定してください。 但し行列の最大非ゼロ要素数は800,000,000以下である必要があります。ここで設定されない行列成分は0となります。 絶対値が1E+20以下の実数を指定してください。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 非ゼロQ_{ij}の数 # number of quadratic terms in objective [変数インデックス] [変数インデックス] [係数値] [変数インデックス] [変数インデックス] [係数値] ……　非ゼロQ_{ij}の数繰り返し　…… [変数インデックス] [変数インデックス] [係数値] 例 2 # number of quadratic terms in objective 2 1 -2 3 2 3.5
number of non-default linear coefficients in objective	default value for linear coefficients in objectiveで設定したBベクトルを上書きします。 上書きしたいベクトル成分の設定値を指定してください。ここで設定されなかった成分にはdefault value for linear coefficients in objectiveタグで設定された値が使用されます。絶対値が1E+20以下の実数を指定してください。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたいBベクトルの成分数 # number of non-default linear coefficients in objective

tag	Description
	[変数インデックス] [係数値] [変数インデックス] [係数値] …… 繰り返し …… [変数インデックス] [係数値] 例 2 # number of non-default linear coefficients in objective 1 -2 2 3.5
number of linear terms in all constraints	A行列を設定します。各行列成分の設定値を指定してください。但しA行列はnumber of constraintsタグで指定した値が行の長さとなり、変数の数が列の長さとなる行列です。 ここで設定されない行列成分は0となります。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 非ゼロA_{ij}の数 # number of linear terms in all constraints [変数インデックス] [変数インデックス] [係数値] [変数インデックス] [変数インデックス] [係数値] …… 非ゼロA_{ij}の数繰り返し …… [変数インデックス] [変数インデックス] [係数値] 例 2 # number of linear terms in all constraints 2 1 -2 3 2 3.5
number of non-default left-hand-sides	default left-hand-side valueで設定したLHSベクトルを上書きします。 上書きしたいベクトル成分の設定値を指定してください。ここで設定されなかった成分にはdefault left-hand-side valueタグで設定された値が使用されます。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたいLHSベクトルの成分数 # number of non-default left-hand-sides [変数インデックス] [制約条件の下限値] [変数インデックス] [制約条件の下限値] …… 繰り返し …… [変数インデックス] [制約条件の下限値] 例 2 # number of non-default left-hand-sides 1 -2 2 3.5
number of non-default right-hand-sides	default right-hand-side valueで設定したRHSベクトルを上書きします。 上書きしたいベクトル成分の設定値を指定してください。ここで設定されなかった成分にはdefault right-hand-side valueタグで設定された値が使用されます。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたいRHSベクトルの成分数 # number of non-default right-hand-sides [変数インデックス] [制約条件の上限値] [変数インデックス] [制約条件の上限値] …… 繰り返し …… [変数インデックス] [制約条件の上限値] 例 2 # number of non-default right-hand-sides 1 -2 2 3.5
number of non-default variable upper bounds	default variable upper bound valueで設定した各変数の定義域上限値を上書きします。 上書きしたい変数の定義域上限値を-1E+20以上の値で指定してください。ここで設定されなかった成分にはdefault variable upper bound valueタグで設定された値が使用されます。但しVariables typeがB(バイナリ) である場合はこのタグを使用しないでください。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたい定義域上限値の数 # number of non-default variable upper bounds

tag	Description
	[変数インデックス] [上限値] [変数インデックス] [上限値] ……　繰り返し　…… [変数インデックス] [上限値] 例 2 # number of non-default variable upper bounds 1 -2 2 3.5
number of non-default variable lower bounds	default variable lower bound valueで設定した各変数の定義域下限値を上書きします。 上書きしたい変数の定義域下限値を1E+20以下の値で指定してください。ここで設定されなかった成分にはdefault variable lower bound valueタグで設定された値が使用されます。但しVariables typeがB(バイナリ) である場合はこのタグを使用しないでください。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたい定義域上限値の数 # number of non-default variable lower bounds [変数インデックス] [下限値] [変数インデックス] [下限値] ……　繰り返し　…… [変数インデックス] [下限値] 例 2 # number of non-default variable lower bounds 1 -2 2 3.5
number of non-default variable types	default variable typeで設定した各変数のデータタイプを上書きします。 上書きしたい変数のデータタイプを指定してください。ここで設定されなかった変数のデータタイプにはdefault variable typeタグで設定された値が使用されます。但しVariables typeがB(バイナリ)、C(連続変数) である場合はこのタグを使用しないでください。 以下のように指定してください。変数インデックスは1からとなり、変数ベクトルの上から順に番号をつけたものです。 上書きしたい変数のデータタイプの数 # number of non-default variable types [変数インデックス] [データタイプ] [変数インデックス] [データタイプ] ……　繰り返し　…… [変数インデックス] [データタイプ] 例 2 # number of non-default variable types 1 0 2 0

入力問題データに関する制限

qplibソルバーが扱える問題サイズには以下の制限があります。

- 変数の数は100,000以下である必要があります。
- 行列の非ゼロ要素数は800,000,000以下である必要があります。
  ただしメモリの関係上800,000,000以下の場合でも解が得られない場合があります。
- 制約数は1,000,000以下である必要があります。

qplibソルバーレスポンス仕様

レスポンスステータス

全ソルバー共通レスポンスステータスと同様です。

レスポンスヘッダー

全ソルバー共通レスポンスヘッダーと同様です。

レスポンスボディ

Table 15の項目を除き、全ソルバー共通レスポンスボディと同様です。

Table 15 qplibソルバーのレスポンスボディ仕様

Property name	Description
value	2次計画問題の目的関数で計算された評価値になります。
result	変数の配列。変数が5つのときには、resultは `[0,1,1,0,0]` にようになります。
param	qplibソルバー固有パラメーターとしてalgo、steps、dt、Cが表示されます。

qplibソルバーの使用例

サンプル問題説明 (ナップサック問題)

0-1ナップサック問題を例にとって、qplibソルバーの使用方法を説明します。0-1ナップサック問題は以下のように定式化されます。

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

v_iはアイテムiの価値であり、x_iはバイナリ変数でアイテムiを選択する(1)か選択しない(0)かを定めています。w_iはアイテムiの重さであり、Wは選択されたアイテムの総重量上限値です。

サンプル問題として、vとw、Wを以下のように設定します。

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

この設定を用いるとナップザック問題を以下のような2次計画問題へと変換することができます。

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

where

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

サンプルqplibフォーマットファイル

前節で扱ったサンプル問題をqplibフォーマットファイルにすると、以下のようになります。

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
```

上記の問題の実行結果は以下のようになります。

> 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}
```


得られた解は$x = [1, 1, 1, 1]$となります。

パラメーターの設定例

パラメーターの設定例と同様です。

ヘルスチェックAPI

ヘルスチェックAPIはSQBM+計算APIの稼働ステータスを返します。

リクエスト仕様

HTTPリクエスト

- 使用するHTTP メソッド: GET
- URL構成: http://{ip}:{port}/healthcheck
  - ip: SQBM+計算APIが稼働しているマシンのIPアドレス
  - port: SQBM+サーバーのポート番号

リクエストヘッダー

設定の必要はありません。

リクエストパラメーター

設定の必要はありません。

レスポンス仕様

レスポンスステータス

もしSQBM+計算APIが正しく稼働しているのならば200を、そうでないならば503を返します。
SQBM＋そのものがダウンしている場合はConnection refusedとなります。

レスポンスヘッダー

Table 16 設定対象レスポンスヘッダーと設定値

Header field	Value
Content-Type	application/health+json

レスポンスボディ

HTTPレスポンスボディはサーバーの状態を示します。

Table 17 レスポンスメッセージ一覧

Status code	Message	Description
200	pass	SQBM+計算APIは正常に稼働しています。
503	fail	SQBM+計算APIは停止しています。

バージョン確認API

バージョン確認APIは現在使用しているSQBM+のバージョンを返します。

リクエスト仕様

HTTPリクエスト

- 使用するHTTP メソッド: GET
- URL構成: http://{ip}:{port}/version
  - ip: SQBM+が稼働しているマシンのIPアドレス
  - port: SQBM+サーバーのポート番号

リクエストヘッダー

設定の必要はありません。

リクエストパラメーター

設定の必要はありません。

レスポンス仕様

レスポンスボディ

HTTPレスポンスボディはSQBM+のバージョンを示します。

Table 18 バージョン確認APIの実行結果

Property name	Description
version	バージョン名

バージョン確認API使用例

```
$ 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-lgU2RFZoSEk+A53QfpzL09hX3u8"
Date: Tue, 28 Sep 2021 00:54:29 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"version":"2.0.0"}
```

システム構成パラメーター

Table 19 に示しているシステム構成パラメーターはSQBM+サーバーの/home/{ユーザーディレクトリ}/configファイルで設定できます。{ユーザーディレクトリ}はAWSの場合はec2-userであり、それ以外の場合はsbm-userです。通常はこれらの値を変える必要はありません。値を変更した場合には、SQBM+サーバーを再起動して変更を反映する必要があります。

Table 19 システム構成パラメーター

Parameter	default	Description
server_port	8000	SQBM+サーバーのポート番号
payload_limit	2000000000	リクエストボディの最大サイズ(バイト)。デフォルトでは2GBです。 もしデフォルト設定値を超える複数リクエストが発行された場合は、 メモリの関係で計算処理は異常終了する可能性があります。場合によっては接続の切断に繋がります。 もしこのようなことが発生した場合は後述のnumber_of_workersを削減してください。 また一度に1リクエストを発行することを推奨します。
timeout_upper_bound	3600	上限計算時間(秒)。リクエスト内のtimeoutパラメーターの値によらずに、 この上限計算時間が経過した際に計算を終了します。
number_of_workers	4	計算リクエストを受け付けるSQBM+サーバーのワーカースレッドの数。
max_requests	20	同時に受付できるリクエスト数。SQBM+計算APIは複数リクエストを受付ますが、 現在それらを並列に処理することはなく1つずつ処理していきます。 サイズが数百MBを超える計算リクエストが複数ある場合には、サーバーエラーを回避するため、 一度に1つリクエストを発行することを検討してください。
default_timeout	10	計算パラメーターtimeoutのデフォルト値
default_maxwait	60	計算パラメーターmaxwaitのデフォルト値
default_maxout	1	計算パラメーターmaxoutのデフォルト値。1以上の整数を指定してください。

メンテナンスとセキュリティ

ログファイル

SQBM＋サーバーの/var/log/sqbm-plus/ディレクトリにログファイルが作成されます。ログファイルを取得する場合にはSQBM+サーバーでないマシンからscpコマンドを使用して収集してください。

Figure 8[5]はAWS環境でec2-userが全ログファイルを収集してlogfiles.tgzとして保存している例を示しています。

```
$ 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
```

Figure 8 ログファイル収集例

ログローテーション

ログローテーション実施条件

ファイルサイズが10MBに到達した場合、ログローテーションを実施します。

ログローテーションの振る舞い

現在のログファイルを＊＊＊.1に変更します。過去のログファイルは、それぞれファイル名の最後の数字を1つインクリメントします。
例：ログファイル名がmain.logの場合、以下のとおりローテートし、main.logは空ファイルとなります。

```
---------------------------------
  ローテート前       ローテート後
---------------------------------
  main.log     ->   main.log.1
  main.log.1   ->   main.log.2
  main.log.2   ->   main.log.3
        :              :
  main.log.9   ->   main.log.10
---------------------------------
```

管理対象のログファイルは最大99世代です。main.logの場合、main.log, main.log.1, ..., main.log.99までの100ファイルになります。ログローテート前にmain.log.99だったファイルはローテート後はファイル削除されます。

SQBM+停止時の対応

SQBM+は停止した際に自動で復旧するようになっています。APIが異常終了した際には、数秒後にヘルスチェックAPIでSQBM+の状態をご確認ください。
ヘルスチェックAPIが応答しない、もしくはレスポンスボディのメッセージがfailである場合はログファイルに従ってログファイルを収集し、SQBM＋がインストールされているサーバーの再起動をしてください。
ヘルスチェックAPIで問題がないにも関わらずSQBM+計算APIの応答がない場合もログファイルに従ってログファイルを収集し、サーバーの再起動をしてください。
再起動しても問題が解決せず問い合わせをする場合は、収集したログファイルのご提供をお願いします。

免責事項

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 ANY CONTENT IN YOUR AWS ACCOUNT, INCLUDING 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.

SQBM+ for On-premises
ユーザーガイド

2023年1月23日　A1版発行　MWS5551A

発　　行　　東芝デジタルソリューションズ株式会社

〒212-8585 神奈川県川崎市幸区堀川町72番地34

---

1. 最適解であることを保証するものではない。 ↩
2. HDFはHDF Groupのトレードマークです。 ↩
3. すべての例において、sqbmplus_serverをご使用のSQBM+計算APIのホスト名に変更に変換してお考え下さい。 ↩
4. ↩
5. private_key.pemをSQBM+サーバーに接続する際に使用するプライベート鍵へのパスに置き換えてください。sqbmplus_serverはご使用のSQBM+サーバーのホスト名に置き換えてください。 ↩