FUJITSU Quantum-inspired Computing

Digital Annealer オンプレミスサービス ユーザーズガイド

第 1 版

P3KD-1312-01

はじめに

本書は、Digital Annealer システムの利用方法および運用管理方法を説明します。

本書の読者

本書は、Digital Annealer システム上で組合せ最適化問題を解くためのプログラムやアプリケーションを開発する方、それらのプログラムやアプリケーションを実行する方を対象として説明します。
本書を読むにあたって、以下の知識が必要です。

- 組合せ最適化問題に関する基本的な知識
- 量子アニーリング、シミュレーテッドアニーリングなどの最適化問題のアルゴリズムに関する基本的な知識
- Web API に関する基本的な知識

本書の構成

本書の構成は以下のとおりです。

● **第 1 章 概要**

Digital Annealer システムの概要、およびシステム構成を説明します。

● **第 2 章 サービスの利用**

サービスを使用して、組合せ最適化問題を解く方法を説明します。

● **付録 A エラーメッセージ**

利用者画面で出力されるエラーメッセージとその対処方法を説明します。

● **用語集**

Digital Annealer システム固有の用語、および関連する用語を説明します。

関連ドキュメント

関連ドキュメントとして、以下のマニュアルがあります。必要に応じて参照してください。

- FUJITSU Quantum-inspired Computing Digital Annealer オンプレミスサービス システム運用管理者ガイド
- FUJITSU Quantum-inspired Computing Digital Annealer On-Premises Service Software License Terms
- FUJITSU Quantum-inspired Computing Digital Annealer オンプレミスサービス 注意事項・制限事項

本書の表記について

本書では、略称および記号を以下のように使用しています。

製品名／技術名の略称について

本書では、製品名／技術名を以下のように表記しています。

製品名／技術名	略称
Digital Annealer Unit	DAU
Google Chrome™ ブラウザ	Google Chrome

記号について

本書では、参照先、キー、メニューなどを表記するために、以下のように記号を使用します。

記号	意味
「　」	本書内の参照先のタイトル、画面での設定値を「　」で囲んでいます
『　』	他マニュアル参照のマニュアル名を『　』で囲んでいます
[　]	画面のボタン名、タブ名、ドロップダウンメニュー、およびキーボードのキー名を示します 例：[設定] ダイアログボックス、[ファイル] メニュー、[項目名]、 [OK] ボタン、[Enter] キー

コマンドインターフェースの説明では、以下のような記号を使用します。

記号	意味
XXXX	値や文字列が可変であることを表す場合、斜体（イタリック体）の文字を使用、または < > で囲みます
<XXXX>	

また、以下のアイコン表記を使用します。

注　意　操作や設定を行ううえで制限される内容や注意が必要な内容が書いてあります。

備 考　操作や設定を行ううえで知っておくと便利な機能や使い方など、本文を補足する内容が書いてあります。

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

高度な安全性が要求される用途への使用について

本サービスは、一般事務用、パーソナル用、家庭用、通常の産業等の一般的用途を想定して開発・設計・製造されているものであり、原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療用機器、兵器システムにおけるミサイル発射制御など、極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、直接生命・身体に対する重大な危険性を伴う用途（以下「ハイセイフティ用途」という）に使用されるよう開発・設計・製造されたものではありません。
お客様は本サービスを必要な安全性を確保する措置を施すことなくハイセイフティ用途に使用しないでください。また、お客様がハイセイフティ用途に本サービスを使用したことにより発生する、お客様または第三者からのいかなる請求または損害賠償に対しても富士通株式会社およびその関連会社は一切責任を負いかねます。

商標について

- Linux® は米国及びその他の国における Linus Torvalds の登録商標です。
- Google は、Google Inc. の商標または登録商標です。
- その他の会社名、各製品名などの固有名詞は、各社の商号、登録商標または商標です。
- その他、会社名、システム名、製品名などには必ずしも商標表示を付記しておりません。

2019 年 2 月 初版

改版履歴表

版数	日付	変更箇所（変更種別）（注）	変更内容
初版	2019 年 2 月	全体	新規作成

注）変更箇所は最新版の項番を示しています。ただし、アスタリスク（*）の付いている項番は旧版の項番を示します。

目次

第１章　概要

この章では、Digital Annealer システムの概要について説明します。

1.1 Digital Annealer とは

Digital Annealer は、量子現象を用いた計算方法に着想を得て設計したデジタル回路です。アニーリング方式を用いて、組合せ最適化問題を高速に解くことができます。従来のコンピュータのようなプログラミングは必要なく、パラメーターを設定するだけで計算が行われます。また、コンピュータの内部で任意の素子同士が自由に信号をやりとりできる全結合型の構造を採用しているため、量子アニーリングマシンに比べ、計算量が膨大で今まで解けなかった複雑な問題を計算させることができます。

Digital Annealer は、QUBO（Quadratic Unconstrained Binary Optimization）を入力データとして持ち、以下の評価関数（エネルギー）を最小化するための組合せを探索します。

$$E(x) = \sum_{i} \sum_{j>i} J_{i,j} x_i x_j + \sum_{i} h_i x_i + c$$

2 値変数 x ($x \in \{0,1\}$) により、組合せの状態を表現しています。

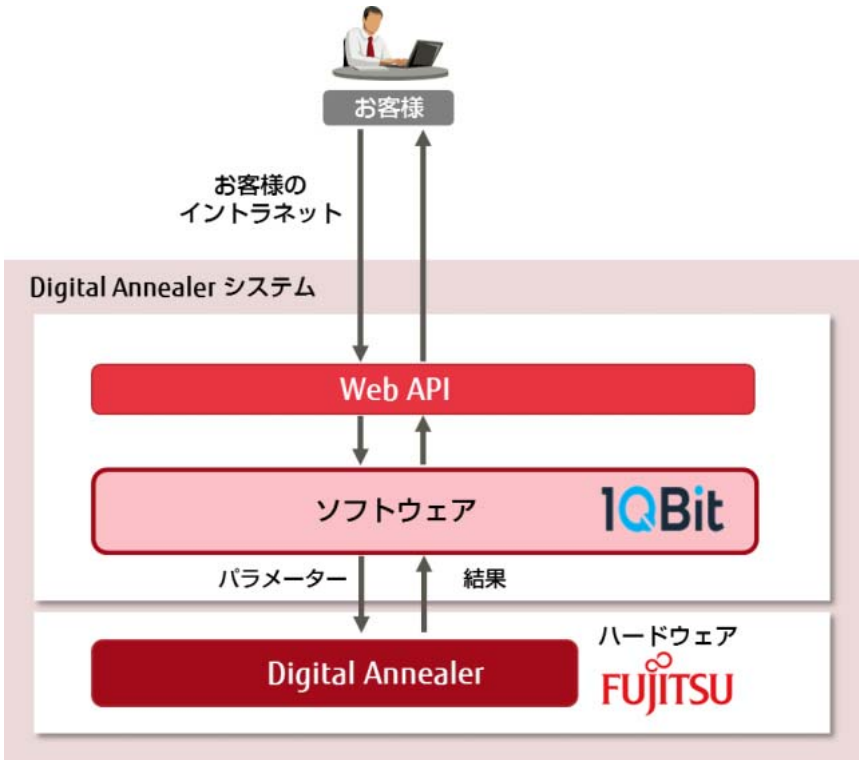
1.1.1 Digital Annealer システムのサービス

Digital Annealer システムは、1QBit 社が開発した量子コンピュータ向けソフトウェア（以降、1QBit ソフトウェア）、および最適化回路 Digital Annealer を実装したハードウェアを使用して、組合せ最適化問題を高速に解くサービスを提供します。

以下のサービスを提供します。

- Web API サービス

図 1：Web API サービス



第 2 章　サービスの利用

この章では、Digital Annealer システムが提供するサービスを使用して、組合せ最適化問題を解く方法について説明します。

2.1 Web API サービス

以下の手順で Web API サービスを使用します。

1. 事前準備
   Web API サービスは、API Gateway 経由で使用できます。
   システム管理者から通知されたユーザー名／パスワードで利用者画面（https（または http）://<IP アドレス >/Userportal/）から Digital Annealer システムにログインし、API Key を作成します。このとき、Web API アクセスのための URL も通知されます。

   注 意

   利用者画面へは、Google Chrome でアクセスしてください。

   備 考

   - 利用者画面を表示する場合に https または http のどちらを指定するかは、システム管理者に確認してください。
   - 利用者画面では、以下の操作が行えます。
     - API Key の作成／表示／削除
     - API アクセス URL の表示
     - パスワードの変更

     利用者画面で表示されるメッセージは、「付録 A エラーメッセージ」(P.37) を参照してください。
   - 利用者画面で API Key を作成したときに表示される Sample Request は、Linux 上で実行する場合の例です。

2. Web API を使用する
   Web API を使用して、組合せ最適化問題を解きます。

2.1.1 Web API の使用方法

Web API の使用方法について説明します。
Web API の使用に関する注意事項は、「2.2.1 Web API 使用時の注意」(P.36) を参照してください。

2.1.1.1 概要

Web API で提供するサービスを以下に示します。

サービス	種別	説明
qubo/hobo2qubo	同期	HOBO（Higher Order Binary Optimization）を QUBO（Quadratic Unconstrained Binary Optimization）に変換する
qubo/solve		QUBO の最適解を求める
async/qubo/solve	非同期	QUBO の最適解を求めるジョブを登録する
async/jobs		ジョブの一覧を取得する
async/jobs/result		ジョブの結果を取得または削除する
async/jobs/cancel		ジョブをキャンセルする

備 考

- qubo/solve は、同一アカウントで最大 5 多重までリクエストできます。
- async/qubo/solve は、同一アカウントで最大 16 多重までリクエストできます。なお、async/qubo/solve の処理結果の数も多重数に含まれるため、不要な処理結果は削除してください。
- 上記以外にも 1QBit 社から Web API サービスが提供されています。
  サービスの種類や仕様については、以下を参照してください。

  http://portal.1qbit.com/documentation

  リクエスト時の URI は、/v1/... ではなく、/v2/... を指定してください。

データ形式と使用するポート番号を以下に示します。

データ形式	ポート番号
REST（REpresentational State Transfer）/JSON 形式	8080

2.1.1.2 Digital Annealer の計算精度

Digital Annealer ハードウェアは、二次のバイナリ多項式の係数を整数としてエンコードします。
Digital Annealer ハードウェアの計算精度は以下のとおりです。最適解を求めるには、QUBO の式の係数を Digital Annealer ハードウェアの計算精度の範囲内で指定する必要があります。

パーティションサイズ（*1）	問題の規模	二次項	線形項
1K	～ 1K	64bit の符号付き整数 (*2)	76bit の符号付き整数
2K	～ 2K	64bit の符号付き整数 (*2)	76bit の符号付き整数
4K	～ 4K	32bit の符号付き整数	76bit の符号付き整数
8K	～ 4K	64bit の符号付き整数 (*2)	76bit の符号付き整数
	4K 超～ 8K	16bit の符号付き整数	76bit の符号付き整数

*1: DAU のパーティションサイズによって精度が異なります。御使用のシステムの設定については、システム管理者に確認してください。
*2: $-2^{63}+1$ ～ $2^{63}-1$ の範囲とします。

[Scaling and Rounding]

QUBO の式の係数が Digital Annealer ハードウェアの計算精度の範囲外や整数でない場合に、Digital Annealer ハードウェア上で解くことができる多項式に自動で変換する機能です。
以下のソルバーを御使用の場合、アニーリング時にこの機能が有効になります。

- 「FujitsuDA2PTSolver」(P.25)
- 「FujitsuDA2Solver」(P.27) を使用し、expert_mode パラメーターに false を指定する場合
- 「FujitsuDA2MixedModeSolver」(P.29)

注 意

qubo/solve で指定可能な QUBO の式の係数の範囲は、「BinaryPolynomial」(P.23) または 「QuboMatrix」(P.24) で確認してください。

2.1.1.3 API の使用例

提供する各 API の使用例を以下に示します。この項の使用例は、Linux 上での実行例です。
各 API の仕様については、「2.1.1.4 API の仕様」(P.16) を参照してください。

2.1.1.3.1 qubo/hobo2qubo の使用例

以下に、qubo/hobo2qubo を使用して、HOBO 形式の数式を QUBO 形式に変換する例を示します。

リクエスト例：$x_1x_2x_3$ を QUBO 形式に変換

1. クライアント上で、以下を実行します。
   <API Key> は、利用者画面で作成した API Key を指定します。
   <Access URL> は、API Key 作成時に通知された API アクセスのための URL を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X POST -d '
{
    "terms": [
        {
            "coefficient": 1.0,
            "polynomials": [ 1, 2, 3 ]
        }
    ]
}
' <Access URL>/v2/qubo/hobo2qubo
```

2. QUBO 形式に変換された結果が Digital Annealer システムから取得できます。
   レスポンス例：$2x_1x_2 - 4x_1x_4 - 4x_2x_4 + x_3x_4 + 6x_4$

```
{"terms":[{"coefficient":2,"polynomials":[1,2]},
{"coefficient":-4,"polynomials":[1,4]},
{"coefficient":-4,"polynomials":[2,4]},
{"coefficient":1,"polynomials":[3,4]},
{"coefficient":6,"polynomials":[4]}]}
```

2.1.1.3.2 qubo/solve の使用例

以下に、qubo/solve を使用して、QUBO の最適解を求める例を示します。

リクエスト例：$2x_1x_2 - 4x_2x_4$ の最適解を求める

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。

   - FujitsuDA2PTSolver 使用の場合

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X POST -d '
{
  "binary_polynomial": {
    "terms": [{
        "coefficient": 2,
        "polynomials": [1, 2]
      },
      {
        "coefficient": -4,
        "polynomials": [2, 4]
      }
    ]
  },
  "fujitsuDA2PT": {
    "number_iterations": 1000000,
    "number_replicas": 26,
    "guidance_config":{"1":false,"2":false,"4":false}
  }
}
' <Access URL>/v2/qubo/solve
```

   - FujitsuDA2Solver 使用の場合

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X POST -d '
{
  "binary_polynomial": {
    "terms": [{
        "coefficient": 2,
        "polynomials": [1, 2]
      },
      {
        "coefficient": -4,
        "polynomials": [2, 4]
      }
    ]
  },
  "fujitsuDA2": {
    "expert_mode": true,
    "number_iterations": 1000000,
    "number_runs": 16,
    "offset_increase_rate": 819,
    "temperature_start": 655,
    "temperature_decay": 0.0001,
    "temperature_mode": "EXPONENTIAL",
    "temperature_interval": 100
  }
}
' <Access URL>/v2/qubo/solve
```

2. QUBO の最適解が Digital Annealer システムから取得できます。
   レスポンス例：FujitsuDA2PTSolver 使用の場合


```
{"qubo_solution":{"result_status":true,"solutions":[{"energy":-4,"frequency":26,
"configuration":{"1":false,"2":true,"4":true}}],
"timing":{"cpu_time":"34","queue_time":"3372","solve_time":"1435","total_elapsed_time":"4807",
"detailed":{"anneal_time":"1401"}}},
"solver_input_parameters":{"guidance_config":"{x1 = 0, x2 = 0, x4 = 0}",
"job_id":"tenant1-183172851152618","number_iterations":"1000000",
"number_replicas":"26","offset_increase_rate":"1000","solution_mode":"COMPLETE",
"solver_name":"FujitsuDA2PT","timeout":"0"}}
```


   レスポンス例：FujitsuDA2Solver 使用の場合


```
{"qubo_solution":{"result_status":true,"solutions":[{"energy":-4,"frequency":16,
"configuration":{"1":false,"2":true,"4":true}}],
"timing":{"cpu_time":"24","queue_time":"3174","solve_time":"77","total_elapsed_time":"3251",
"detailed":{"anneal_time":"53"}}},
"solver_input_parameters":{"expert_mode":"true","job_id":"tenant1-183172900860516",
"number_iterations":"1000000","number_runs":"16",
"offset_increase_rate":"819","solution_mode":"COMPLETE","solver_name":"FujitsuDA2",
"temperature_decay":"0.0001",
"temperature_interval":"100","temperature_mode":"EXPONENTIAL","temperature_start":"655",
"timeout":"0"}}
```


2.1.1.3.3 async/qubo/solve の使用例

以下に、async/qubo/solve を使用して、QUBO の最適解を求めるジョブを登録する例を示します。

リクエスト例：$2x_1x_2 - 4x_2x_4$ の最適解を求めるジョブを登録

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X POST -d '
{
  "binary_polynomial": {
    "terms": [{
        "coefficient": 2,
        "polynomials": [1, 2]
      },
      {
        "coefficient": -4,
        "polynomials": [2, 4]
      }
    ]
  },
  "fujitsuDA2PT": {
    "number_iterations": 1000000,
    "number_replicas": 26,
    "guidance_config":{"1":false,"2":false,"4":false}
   }
}
' <Access URL>/v2/async/qubo/solve
```

2. 登録されたジョブ番号（job_id）が Digital Annealer システムから取得できます。
   レスポンス例：


```
  {"job_id":"ABC00001-1234512345123"}
```

2.1.1.3.4 async/jobs の使用例

以下に、async/jobs を使用して、登録されたジョブの一覧を取得する例を示します。

リクエスト例：

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X GET <Access URL>/v2/async/jobs
```

2. 登録されているすべてのジョブ番号（job_id）の一覧が Digital Annealer システムから取得できます。
   レスポンス例：

```
{"job_status_list":[
{"job_id":"tenant1-181910349357","job_status":"Done","start_time":"2018-07-
10T06:02:22Z"},
{"job_id":"tenant1-181910350804","job_status":"Running","start_time":"2018-07-
10T06:02:24Z"},
{"job_id":"tenant1-181910633669","job_status":"Error","start_time":"2018-07-
10T06:05:07Z"},
{"job_id":"tenant1-181910648337","job_status":"Waiting","start_time":"2018-07-
10T06:05:21Z"}]}
```

2.1.1.3.5 async/jobs/result の使用例

以下に、async/jobs/result を使用して、ジョブの結果を取得または削除する例を示します。

リクエスト例 1：ジョブの結果（QUBO の最適解）を取得する

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。
   <*Job ID*> は、登録されているジョブ番号を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X GET <Access URL>/v2/async/jobs/result/<Job ID>
```

2. QUBO の最適解が Digital Annealer システムから取得できます。
   レスポンス例 1：ジョブが完了している場合

```
{"qubo_solution":{"result_status":true,"solutions":[{"energy":-4,"frequency":26,
"configuration":{"1":false,"2":true,"4":true}}],
"timing":{"cpu_time":"34","queue_time":"3372","solve_time":"1435","total_elapsed_time":"4807",
"detailed":{"anneal_time":"1401"}}},
"solver_input_parameters":{"guidance_config":"{x1 = 0, x2 = 0, x4 = 0}",
"job_id":"tenant1-183172851152618","number_iterations":"1000000",
"number_replicas":"26","offset_increase_rate":"1000","solution_mode":"COMPLETE",
"solver_name":"FujitsuDA2PT","timeout":"0"},{"status":"Done"}}
```

   レスポンス例 2：ジョブが実行中の場合

```
{"status":"Running"}
```

   レスポンス例 3：ジョブが実行待ちの場合

```
{"status":"Waiting"}
```

リクエスト例 2：ジョブの結果を削除する

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。
   <*Job ID*> は、登録されているジョブ番号を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X DELETE <Access URL>/v2/async/jobs/result/<Job ID>
```

2. ジョブが完了している場合、ジョブの結果が Digital Annealer システムから削除されます。
   レスポンス例 1：ジョブが完了しており、削除された場合

```
{"qubo_solution":{"result_status":true,"solutions":[{"energy":-4,"frequency":26,
"configuration":{"1":false,"2":true,"4":true}}],
"timing":{"cpu_time":"34","queue_time":"3372","solve_time":"1435","total_elapsed_time":"4807",
"detailed":{"anneal_time":"1401"}}},
"solver_input_parameters":{"guidance_config":"{x1 = 0, x2 = 0, x4 = 0}",
"job_id":"tenant1-183172851152618","number_iterations":"1000000",
"number_replicas":"26","offset_increase_rate":"1000","solution_mode":"COMPLETE",
"solver_name":"FujitsuDA2PT","timeout":"0"},{"status":"Deleted"}}
```

   レスポンス例 2：ジョブが実行中の場合

```
{"status":"Running"}
```

   レスポンス例 3：ジョブが実行待ちの場合

```
{"status":"Waiting"}
```

2.1.1.3.6 async/jobs/cancel の使用例

以下に、async/jobs/cancel を使用して、ジョブをキャンセルする例を示します。

リクエスト例：

1. クライアント上で、以下を実行します。
   <*API Key*> は、利用者画面で作成した API Key を指定します。
   <*Access URL*> は、API Key 作成時に通知された API アクセスのための URL を指定します。
   <*Job ID*> は、登録されているジョブ番号を指定します。

```
$ curl -H 'X-DA-Access-Key:<API Key>' -H 'Accept: application/json' -H 'Content-type
: application/json' -X POST -d '
{
   "job_id" :"<Job ID>"
}
' <Access URL>/v2/async/jobs/cancel
```

2. ジョブが実行待ちの場合、ジョブの実行が Digital Annealer システムからキャンセルされます。
   レスポンス例 1：ジョブが実行待ちでありキャンセルされた場合

```
{"status":"Canceled"}
```

   レスポンス例 2：ジョブが完了しているためキャンセルできなかった場合

```
{"status":"Done"}
```

   レスポンス例 3：ジョブが実行中のためキャンセルできなかった場合

```
{"status":"Running"}
```

2.1.1.4 API の仕様

提供する各 API の仕様とデータ形式を以下に示します。

2.1.1.4.1 qubo/hobo2qubo の仕様

qubo/hobo2qubo の仕様を以下に示します。

- メソッド種別
  POST
- URI
  /v2/qubo/hobo2qubo
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  BinaryPolynomial
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  BinaryPolynomial
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200：　　正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。

2.1.1.4.2 qubo/solve の仕様

qubo/solve の仕様を以下に示します。

- メソッド種別
  POST
- URI
  /v2/qubo/solve
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  QuboRequest
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  QuboResponse
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200： 正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。
  Digital Annealer ハードウェアの処理で異常が検出された場合は、以下のようなメッセージが通知されます。

```
Digital Annealer failed to do the anneal with code: -1
```

【原因】
指定したパラメーターに誤りがあるなど、アプリケーションのエラーであることを示します。

【対処】
指定したパラメーターに誤りがないかなど確認してください。

```
Digital Annealer failed to do the anneal with code: -2
```

【原因】
システムのエラーであることを示します。

【対処】
サポート窓口へお問合せください。

`Digital Annealer failed to do the anneal with code: -3`

【原因】

資源が一時的に利用できないなど、リトライ可能なエラーであることを示します。

【対処】

しばらく時間をおいてから再試行してください。

`Digital Annealer failed to do the anneal with code: -4`

【原因】

ハードウェアのエラーが発生したことを示します。

【対処】

サポート窓口へお問合せください。

`Digital Annealer failed to do the anneal with code: -6`

【原因】

Job ID に関するエラーが発生したことを示します。

【対処】

リクエストボディに複数のソルバーの設定がないか、または JSON フォーマットの異常がないかを確認してください。

`Digital Annealer failed to do the anneal with code: -7`

【原因】

温度が負の値になる場合など、浮動小数点演算のエラーが発生したことを示します。

【対処】

温度関連パラメーターの設定に誤りがないかなどを確認してください。

2.1.1.4.3 async/qubo/solve の仕様

async/qubo/solve の仕様を以下に示します。

- メソッド種別
  POST
- URI
  /v2/async/qubo/solve
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  QuboRequest
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  JobID
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200：　　正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。

2.1.1.4.4 async/jobs の仕様

async/jobs の仕様を以下に示します。

- メソッド種別
  GET
- URI
  /v2/async/jobs
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  なし
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  JobStatusList
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200：　　正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。

2.1.1.4.5 async/jobs/result の仕様

async/jobs/result の仕様を以下に示します。

- メソッド種別
  GET / DELETE
- URI
  /v2/async/jobs/result/<*Job ID*> （*1）

  *1: <*Job ID*> は、登録されているジョブ番号を指定します。
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  なし
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  - ジョブが完了している場合
    QuboResponse
  - ジョブが完了していない場合
    JobStatus
    データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200： 正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。
  Digital Annealer ハードウェアの処理で異常が検出された場合に通知されるメッセージは、「2.1.1.4.2 qubo/solve の仕様」(P.17) を参照してください。

2.1.1.4.6 async/jobs/cancel の仕様

async/jobs/cancel の仕様を以下に示します。

- メソッド種別
  POST
- URI
  /v2/async/jobs/cancel
- リクエストヘッダーの説明

キー	説明
Content-Length	<length> リクエストボディのサイズ（単位：バイト）
Content-Type	application/json（固定）
Accept	application/json（固定）
X-DA-Access-Key	利用者画面で取得した API Key

- リクエストボディの形式
  JobID
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- レスポンスボディの形式
  JobStatus
  データ形式は、「2.1.1.5 API のデータ形式」(P.23) を参照してください。
- 処理結果
  処理結果は、HTTP ステータスコードで返されます。
  200：　　正常
  200 以外：異常

  異常の内容は、通知されるエラーメッセージで確認してください。

2.1.1.5 API のデータ形式

提供する各 API のリクエストおよびレスポンスのデータ形式を以下に示します。

BinaryPolynomial

```
BinaryPolynomial{
   terms [BinaryPolynomialTerm]
}
```

キー	説明
terms	BinaryPolynomialTerm の配列

```
BinaryPolynomialTerm{
                    coefficient  number($double)
                    polynomials  [integer($uint32)]
}
```

キー	説明
coefficient	係数（double 型） 値は以下の範囲で指定 • 問題の規模が 4K 超～ 8K の場合 -2^{15} = -32,768 ～ 2^{15} -1 = 32,767 の値を指定 • パーティションサイズが 4K の場合 -2^{31} = -2,147,483,648 ～ 2^{31} -1 = 2,147,483,647 の値を指定 • 上記以外 -2^{62} = -4,611,686,018,427,387,904 ～ 2^{62} = 4,611,686,018,427,387,904 の値を指定 ただし、double 型（倍精度浮動小数点数）の丸め誤差の影響を受けないようにするため、以下の範囲の値を指定することを推奨 -2^{53} = -9,007,199,254,740,992 ～ 2^{53} = 9,007,199,254,740,992
polynomials	変数の番号の配列（uint32 型）

指定例：$2x_1x_2 - 4x_2x_4 + x_4 + 3$

```
{
    "terms": [
        {
            "coefficient": 2,
            "polynomials": [ 1, 2 ]
        },
        {
            "coefficient": -4,
            "polynomials": [ 2, 4 ]
        },
        {
            "coefficient": 1,
            "polynomials": [ 4 ]
        },
        {
            "coefficient": 3
        }
    ]
}
```

2 次多項式の各項単位に、coefficient に係数や定数を指定し、polynomials に変数の番号をカンマで区切って指定します。

注　意

qubo/solve で、ソルバーに FujitsuDA2Solver を使用する場合、coefficient に指定する係数は「2.1.1.2 Digital Annealer の計算精度」(P.10) に従い、ハードウェア上で計算できる範囲の整数で指定してください。

QuboMatrix

```
QuboMatrix {
  qubo           [QuboMatrixQuboArray]
}
```

キー	説明
qubo	QuboMatrixQuboArray の配列

```
QuboMatrixQuboArray {
  qubo_row     [number($double)]
}
```

キー	説明
qubo_row	qubo の行の配列（double 型） 値は以下の範囲で指定 • 問題の規模が 4K 超～ 8K の場合 -2^{15} = -32,768 ～ 2^{15} -1 = 32,767 の値を指定 • パーティションサイズが 4K の場合 -2^{31} = -2,147,483,648 ～ 2^{31} -1 = 2,147,483,647 の値を指定 • 上記以外 -2^{62} = -4,611,686,018,427,387,904 ～ 2^{62} = 4,611,686,018,427,387,904 の値を指定 ただし、double 型（倍精度浮動小数点数）の丸め誤差の影響を受けないようにするため、以下の範囲の値を指定することを推奨 -2^{53} = -9,007,199,254,740,992 ～ 2^{53} = 9,007,199,254,740,992

指定例：$2x_1x_2 - 4x_2x_4 + x_4$

```
{
    "qubo": [
        {
            "qubo_row": [ 0, 0, 0, 0, 0 ]
        },
        {
            "qubo_row": [ 0, 0, 2, 0, 0 ]
        },
        {
            "qubo_row": [ 0, 0, 0, 0, -4 ]
        },
        {
            "qubo_row": [ 0, 0, 0, 0, 0 ]
        },
        {
            "qubo_row": [ 0, 0, 0, 0, 1 ]
        }
    ]
}
```

2 次多項式の各項の 1 つ目の変数と 2 つ目の変数を、列と行でそれぞれ以下のように表し、それぞれの変数の番号の要素に係数を指定します。

列 $\{x_0,x_1,x_2,\ldots x_n\}$
行 $\{x_0,x_1,x_2,\ldots x_n\}$

1 次式の場合は、変数の番号の対角項に係数を指定します。
定数は指定できないため、求められた最適解に加算減算するか、または BinaryPolynomial で指定してください。

注　意

qubo/solve で、ソルバーに FujitsuDA2Solver を使用する場合、qubo_row に指定する係数は「2.1.1.2 Digital Annealer の計算精度」(P.10) に従い、ハードウェア上で計算できる範囲の整数で指定してください。

QuboRequest

```
QuboRequest {
  binary_polynomial             BinaryPolynomial           (*1)
  qubo_matrix                   QuboMatrix                 (*1)

  fujitsuDA2PT                  FujitsuDA2PTSolver         (*2)
  fujitsuDA2                    FujitsuDA2Solver           (*2)
  fujitsuDA2MixedMode           FujitsuDA2MixedModeSolver  (*2)
}
```

*1:　2 種類の形式（BinaryPolynomial / QuboMatrix）のどちらかで QUBO を指定します。
*2:　3 種類のソルバー（FujitsuDA2PTSolver / FujitsuDA2Solver / FujitsuDA2MixedModeSolver）のどれかを指定します。レプリカ交換（parallel tempering）機能を使用する場合は、FujitsuDA2PTSolver を指定してください。
レプリカ交換機能については、「備考」(P.31) および「レプリカ交換」(P.39) の説明を参照してください。

ソルバー（FujitsuDA2PTSolver / FujitsuDA2Solver / FujitsuDA2MixedModeSolver）のパラメーターの詳細を以下に示します。

● FujitsuDA2PTSolver

初回実行は、Scaling and Rounding（「2.1.1.2 Digital Annealer の計算精度」(P.10) を参照）が有効であり、かつ、温度スケジュールに関わるパラメーターのチューニング作業が不要（「備考」(P.31) の説明を参照）である、FujitsuDA2PTSolver の指定を推奨します。
FujitsuDA2PTSolver を指定しても、最適解を得るまで時間がかかる場合、または最適解が得られない場合は、「FujitsuDA2Solver」(P.27) または「FujitsuDA2MixedModeSolver」(P.29) を指定した実行をお試しください。

```
FujitsuDA2PTSolver {
   number_iterations              integer($int32)
   number_replicas                integer($int32)
   offset_increase_rate           number($float)
   solution_mode                  string
   guidance_config                {"integer($uint32)":boolean($boolean)}
}
```

パラメーター	説明
number_iterations	1 回のアニーリングにおける探索回数（int32 型） 1 ～ 2000000000 の整数を指定（デフォルト：2000000）（推奨値：1000000 以上）

パラメーター	説明
number_replicas	レプリカ数（int32 型） 初期化された異なる温度で並列にアニーリングを実行させる数 26 ～ 128 の整数を指定 設定値に制限があるため、「2.2.1 Web API 使用時の注意」(P.36) を参照
offset_increase_rate	新しい状態が選ばれなかったときのエネルギー累積増分（float 型） 本パラメーターの指定は不要 指定した場合、値は無効
solution_mode	最適解の復帰モード（string 型） COMPLETE または QUICK を指定（デフォルト：COMPLETE） • COMPLETE を指定した場合 number_replicas で指定したアニーリング数分すべての結果を返すが、 同一の結果は 1 つにまとめ、frequency に合計した出現頻度を設定して返す • QUICK を指定した場合 number_replicas で指定したアニーリング数分すべての結果の中から、 energy が一番低い結果（最適解）を 1 つだけ返す
guidance_config	各変数の初期値（"uint32 型 ":boolean 型） 最適解を求める際に設定する、多項式（問題）の各変数の初期値を指定 最適解に近い値を指定することで、最適解の確度の向上が期待できる。また、 指定した初期値で同じ多項式（問題）を繰り返し解いた場合、毎回同じ最適解 が得られる 以下の形式で、変数の数分の初期値を指定する 形式：　{" 変数番号 ": 初期値 , " 変数番号 ": 初期値 , ･･･ } 指定例：$2x_1x_2 - 4x_2x_4$ の各変数に初期値を指定する場合 {"1":false,"2":false,"4":false} 指定しない場合は、ソルバーがランダムに設定

● FujitsuDA2Solver

```
FujitsuDA2Solver {
   expert_mode                  boolean($boolean)
   number_iterations            integer($int32)
   number_runs                  integer($int32)
   offset_increase_rate         number($float)
   temperature_decay            number($float)
   temperature_interval         integer($int32)
   temperature_mode             string
   temperature_start            number($float)
   solution_mode                string
   guidance_config              {"integer($uint32)":boolean($boolean)}
}
```

パラメーター	説明
expert_mode	エキスパートモード（boolean 型） アニーリングを行うためのスケーリング（Scaling and Rounding）、 および各種パラメーター（Parameter Setting）を手動で設定するか 自動で設定するかを、true または false で指定（デフォルト：false） • true を指定した場合 Scaling and Rounding：disable（手動） Parameter Setting : enable（手動） • false を指定した場合 Scaling and Rounding：enable（自動） Parameter Setting : disable（自動） 以下の設定にする場合は、「FujitsuDA2MixedModeSolver」(P.29) を指定 Scaling and Rounding：enable（自動） Parameter Setting：enable（手動） [Scaling and Rounding] 「2.1.1.2 Digital Annealer の計算精度」(P.10) を参照 [Parameter Setting] 以下のパラメーターは enable の場合だけ指定可能 • offset_increase_rate • temperature_decay • temperature_interval • temperature_mode • temperature_start
number_iterations	1 回のアニーリングにおける探索回数（int32 型） 1 ～ 2000000000 の整数を指定
number_runs	アニーリングの繰り返し回数（int32 型） 16 ～ 128 の整数を指定 設定値に制限があるため、「2.2.1 Web API 使用時の注意」(P.36) を参照
offset_increase_rate (*1)	新しい状態が選ばれなかったときのエネルギー累積増分（float 型） 0 ～ 2000000000 の値を指定
temperature_decay (*1)	アニーリングの温度の減衰率（float 型） 指定値の範囲は、temperature_mode の指定に依存 「temperature_mode」(P.28) の説明を参照
temperature_interval (*1)	アニーリング時の温度変更の間隔（int32 型） 1 以上の整数を指定

パラメーター	説明
temperature_mode （*1）	アニーリング温度の変更モデル（string 型） EXPONENTIAL、INVERSE、または INVERSE_ROOT のどれかを指定（デフォルト：EXPONENTIAL） EXPONENTIAL を推奨 *n* 番目の温度変更時における温度（T_n）は以下のように算出する • EXPONENTIAL $T_{n+1} = T_n \times$（1 - *temperature_decay*） • INVERSE $T_{n+1} = T_n \times$（1 - *temperature_decay* × T_n） • INVERSE_ROOT $T_{n+1} = T_n \times$（1 - *temperature_decay* × T_n × T_n） temperature_decay は、各モードの定義式の右辺が 0 以上の値になるように指定 各モードの指定値の範囲は以下 • EXPONENTIAL 0 ≦ temperature_decay ≦ 1 • INVERSE 0 ≦ temperature_decay ≦ 1 / temperature_start • INVERSE_ROOT 0 ≦ temperature_decay ≦ 1 /（ temperature_start × temperature_start） temperature_decay の値を小さくすると、アニーリングの温度遷移は緩やかになる
temperature_start （*1）	アニーリングの開始温度（float 型） 0 より大きな値を指定
solution_mode	最適解の復帰モード（string 型） COMPLETE または QUICK を指定（デフォルト：COMPLETE） • COMPLETE を指定した場合 number_runs で指定したアニーリング数分すべての結果を返すが、 同一の結果は 1 つにまとめ、frequency に合計した出現頻度を設定して返す • QUICK を指定した場合 number_runs で指定したアニーリング数分すべての結果の中から、 energy が一番低い結果（最適解）を 1 つだけ返す
guidance_config	各変数の初期値（"uint32 型 ":boolean 型） 最適解を求める際に設定する、多項式（問題）の各変数の初期値を指定 最適解に近い値を指定することで、最適解の確度の向上が期待できる。また、指定した初期値で同じ多項式（問題）を繰り返し解いた場合、毎回同じ最適解が得られる 以下の形式で、変数の数分の初期値を指定する 形式：　{" 変数番号 ": 初期値 , " 変数番号 ": 初期値 , ･･･ } 指定例：$2x_1x_2 - 4x_2x_4$ の各変数に初期値を指定する場合 {"1":false,"2":false,"4":false} 指定しない場合は、ソルバーがランダムに設定

*1:　expert_mode パラメーターで true を指定した場合だけ、指定できます。

● FujitsuDA2MixedModeSolver

Scaling and Rounding（「2.1.1.2 Digital Annealer の計算精度」(P.10) を参照）が有効で、各種温度パラメーターを手動で設定する場合に指定します。

```
FujitsuDA2MixedModeSolver {
   number_iterations               integer($int32)
   number_runs                     integer($int32)
   offset_increase_rate            number($float)
   temperature_decay               number($float)
   temperature_interval            integer($int32)
   temperature_mode                string
   temperature_start               number($float)
   solution_mode                   string
   guidance_config                 {"integer($uint32)":boolean($boolean)}
}
```

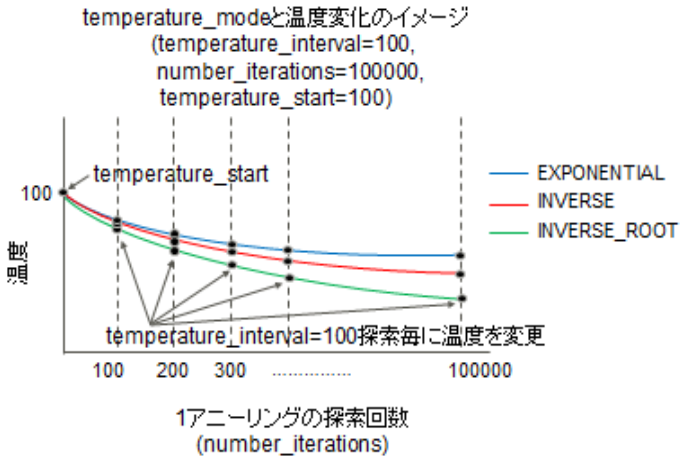
パラメーター	説明
number_iterations	1 回のアニーリングにおける探索回数（int32 型） 1 ～ 2000000000 の整数を指定
number_runs	アニーリングの繰り返し回数（int32 型） 16 ～ 128 の整数を指定 設定値に制限があるため、「2.2.1 Web API 使用時の注意」(P.36) を参照
offset_increase_rate	新しい状態が選ばれなかったときのエネルギー累積増分（float 型） 0 ～ 2000000000 の値を指定
temperature_decay	アニーリングの温度の減衰率（float 型） 指定値の範囲は、temperature_mode の指定に依存 「temperature_mode」(P.29) の説明を参照
temperature_interval	アニーリング時の温度変更の間隔（int32 型） 1 以上の整数を指定
temperature_mode	アニーリング温度の変更モデル（string 型） EXPONENTIAL、INVERSE、または INVERSE_ROOT のどれかを指定（デフォルト：EXPONENTIAL） EXPONENTIAL を推奨 n 番目の温度変更時における温度（Tn）は以下のように算出する • EXPONENTIAL $Tn+1 = Tn \times (1 - temperature\_decay)$ • INVERSE $Tn+1 = Tn \times (1 - temperature\_decay \times Tn)$ • INVERSE_ROOT $Tn+1 = Tn \times (1 - temperature\_decay \times Tn \times Tn)$ temperature_decay は、各モードの定義式の右辺が 0 以上の値になるように指定 各モードの指定値の範囲は以下 • EXPONENTIAL 0 ≦ temperature_decay ≦ 1 • INVERSE 0 ≦ temperature_decay ≦ 1 / temperature_start • INVERSE_ROOT 0 ≦ temperature_decay ≦ 1 /（ temperature_start × temperature_start） temperature_decay の値を小さくすると、アニーリングの温度遷移は緩やかになる
temperature_start	アニーリングの開始温度（float 型） 0 より大きな値を指定

パラメーター	説明
solution_mode	最適解の復帰モード（string 型） COMPLETE または QUICK を指定（デフォルト：COMPLETE） • COMPLETE を指定した場合 number_runs で指定したアニーリング数分すべての結果を返すが、 同一の結果は 1 つにまとめ、frequency に合計した出現頻度を設定して返す • QUICK を指定した場合 number_runs で指定したアニーリング数分すべての結果の中から、 energy が一番低い結果（最適解）を 1 つだけ返す
guidance_config	各変数の初期値（"uint32 型 ":boolean 型） 最適解を求める際に設定する、多項式（問題）の各変数の初期値を指定 最適解に近い値を指定することで、最適解の確度の向上が期待できる。また、 指定した初期値で同じ多項式（問題）を繰り返し解いた場合、毎回同じ最適解 が得られる 以下の形式で、変数の数分の初期値を指定する 形式：　{" 変数番号 ": 初期値 , " 変数番号 ": 初期値 , ･･･ } 指定例：$2x_1x_2 - 4x_2x_4$ の各変数に初期値を指定する場合 {"1":false,"2":false,"4":false} 指定しない場合は、ソルバーがランダムに設定

◯ 備 考

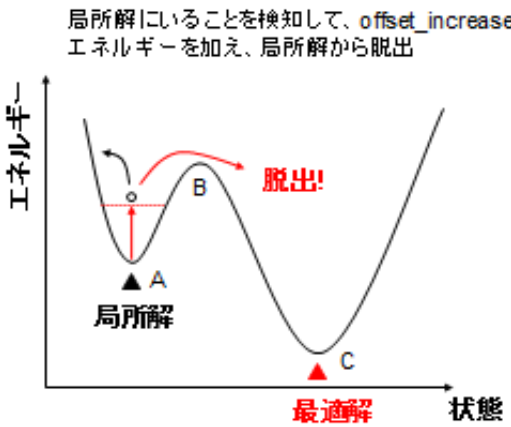
- FujitsuDA2PTSolver（レプリカ交換機能使用）では、QUBO の最適解を求めるために必要な以下のパラメーターの指定が不要になります。
  - temperature_decay
  - temperature_interval
  - temperature_mode
  - temperature_start

  FujitsuDA2Solver / FujitsuDA2MixedModeSolver では、特に temperature_decay、temperature_interval、temperature_start は、温度スケジュールに関わるパラメーターであり、適切な値を指定しないと最適解が得られません。
  各パラメーターの値の組合せは無数にあり、最適解が得られる適切な値を探すチューニング作業が必要となります。
  FujitsuDA2PTSolver では、これらのパラメーターの指定が不要となり、チューニング作業にかかる時間を大幅に短縮できます。
- FujitsuDA2Solver / FujitsuDA2MixedModeSolver では、以下のパラメーターのうち、どれか 1 つでも指定する場合は、必ず以下の 5 つのパラメーターおよび number_iterations パラメーターをすべて指定してください。
  - offset_increase_rate
  - temperature_decay
  - temperature_interval
  - temperature_mode
  - temperature_start
- FujitsuDA2Solver / FujitsuDA2MixedModeSolver でのアニーリングの温度スケジュールは、temperature_decay、temperature_interval、temperature_mode、temperature_start のパラメーターで規定されます。
  例えば、以下のように number_iterations=100000、temperature_interval=100 の場合、100 回の探索ごとに temperature_mode パラメーターで指定したモードで算出した温度に変更します。
  1 回のアニーリングの探索回数（number_iterations）が 100000 なので、100000/100=1000 回の温度変更をすることになります。



- FujitsuDA2Solver / FujitsuDA2MixedModeSolver での 1 回のアニーリングは、上記アニーリングの温度スケジュールで設定された各温度で、number_iterations パラメーターで指定した回数分探索が実施され、number_runs パラメーターで指定した回数分アニーリングを繰り返します。
  どちらのパラメーターも大きな値を指定するほど時間がかかります。

- offset_increase_rate は、探索を加速するためのパラメーターです。局所解に落ち込んで状態遷移が起こらない場合に、探索ごとに offset_increase_rate の値分ずつエネルギーを上げる処理を行い、状態遷移を起こす確率を向上させます。状態遷移が発生したら、offset_increase_rate によって累積されたエネルギーはリセットされます。使用しない場合は、0 を指定します。



- QuboRequest のソルバーには、1QBit 社提供のソルバーも指定できます。
  ソルバーの種類や仕様については、以下を参照してください。
  http://portal.1qbit.com/documentation
  リクエスト時の URI は、/v1/... ではなく、/v2/... を指定してください。

QuboResponse

```
QuboResponse {
  qubo_solution QuboSolutionList
  solver_input_parameters {string:string
  {
    status  string (*1)
  }
}
```

キー	説明
qubo_solution	qubo_solution リスト
solver_input_parameters	ソルバーに設定されている各パラメーター名と値 値を指定しなかったパラメーターは、そのパラメーターのデフォルトの値、または指定範囲外の値（0 または -1）が表示される
status（*1）	ジョブの状態 "Done"（完了）または "Delete"（削除）が表示される

*1:　async/job/result サービスだけ表示されます。

```
QuboSolutionList {
  result_status  boolean($boolean)
  solutions      [QuboSolution]
  timing         SolverTiming
}
```

キー	説明
result_status	処理結果のステータス（true または false）
solutions	QuboSolution の配列
timing	timing リスト

```
QuboSolution {
  energy             number($double)
  frequency          number($double)
  configuration {     < * > : boolean($boolean)
  }
}
```

キー	説明
energy	最適解
frequency	最適解の出現頻度
configuration	各変数 X の値（true または false）

注　意

energy は、-2^{53} = -9,007,199,254,740,992 ～ 2^{53} = 9,007,199,254,740,992 の範囲外であった場合、double 型（倍精度浮動小数点数）の丸め誤差の影響により、近似解になることがあります。

```
SolverTiming {
  cpu_time            integer($uint64)
  queue_time          integer($uint64)
  solve_time          integer($uint64)
  total_elapsed_time  integer($uint64)
  detailed            ExtraInfo
}
```

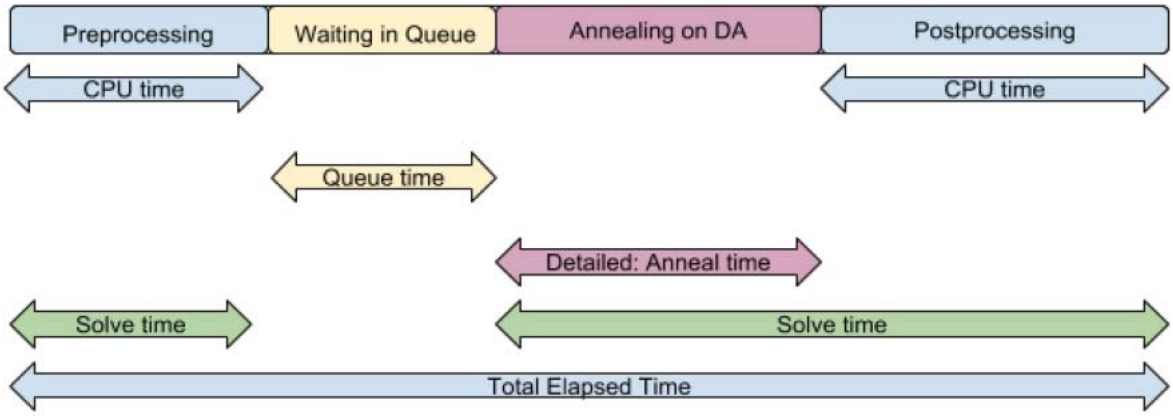
キー	説明
cpu_time	CPU 使用時間（単位：ms）
queue_time	処理待ち時間（単位：ms）
solve_time	ソルバーの処理時間 （単位：ms）（*1）
total_elapsed_time	最適解を求めるのに要したトータル時間（単位：ms）
detailed	detailed リスト

*1:　ほかの処理により待たされた時間も含みます。

```
ExtraInfo {
  anneal_time          integer($uint64)
}
```

キー	説明
anneal_time	Digital Annealer ハードウェアでの処理時間（単位：ms）

図 2：SolverTiming の内訳



JobID

```
JobID {
  job_id  string
}
```

キー	説明
job_id	ジョブ番号

JobStatusList

```
JobStatusList {
  job_status_list  [JobStatusArray]
}
```

キー	説明
job_status_list	JobStatusArray の配列

```
JobStatusArray {
  job_id   string
  job_status  string
  start_time  string
}
```

キー	説明
job_id	ジョブ番号
job_status	ジョブの状態（*1）
start_time	ジョブが登録された UTC 時間

*1:　状態の種類は、「JobStatus」(P.35) を参照。

注　意

各キーの並び順は固定ではありません。

JobStatus

```
JobStatus {
  status  string
}
```

キー	説明
status	ジョブの状態が、以下のどれかで表示される "Done"　　：ジョブが完了していることを示す "Running"　：ジョブが実行中であることを示す（*1） "Waiting"　：ジョブが実行待ちであることを示す "Canceled"：ジョブがキャンセルされたことを示す "Error"　　：ジョブが異常終了したことを示す

*1:　CPU または Digital Annealer ハードウェアで処理を実行している場合に表示されます。

2.2 注意事項

2.2.1 Web API 使用時の注意

Web API 使用時の注意事項について説明します。

- API は、リクエスト後に中断することはできません。
  間違ってリクエストしてしまったり、処理に時間がかかったりしている場合でも、処理が終了するまでお待ちください。
  ただし、ジョブが実行前であれば、キャンセル（「2.1.1.3.6 async/jobs/cancel の使用例」(P.15) 参照）できます。
- Digital Annealer ハードウェアがアニーリング処理中に、qubo/solve の新たなリクエストを発行した場合、実行中のアニーリング処理の完了を待ち合わせます。
  実行中のアニーリング処理完了後に、新たなリクエストの処理が開始されます。
- アニーリングにかかる時間（anneal_time）は、number_iteration パラメーター、number_replicas パラメーター（FujitsuDA2PTSolver の場合）、number_runs パラメーター（FujitsuDA2Solver / FujitsuDA2MixedModeSolver の場合）に指定する値で決まります。例えば、number_iterations = 10000000、number_runs = 100 を指定した場合、20 秒程度かかります。
  値を 10 倍、100 倍 ... すると、anneal_time もそれに比例して 10 倍、100 倍 ... に増加するため、注意して指定してください。
- 問題の規模（ビット数）が大きいと、スケーリング処理やエネルギーの再計算など、CPU を使用して計算する処理に時間がかかることがあります。指定したソルバーやパラメーターの設定値によって、CPU を使用して計算する処理にかかる時間は異なります。
- FujitsuDA2PTSolver を使用した場合、問題の種類や難易度によっては、anneal_time が FujitsuDA2Solver / FujitsuDA2MixedModeSolver の 10 倍程度長くかかることがあります。
- 以下のパラメーターに、上限値および下限値が設定されています。

ソルバー	パラメーター	上限値		下限値
		同期型	非同期型	
FujitsuDA2PTSolver	number_replicas	128	128	26
	number_replicas × number_iterations	100000000000	256000000000	100000
FujitsuDA2Solver / FujitsuDA2MixedModeSolver	number_runs	128	128	16
	number_runs × number_iterations	100000000000	256000000000	100000

- リクエスト時に、キーやパラメーターの指定誤り（スペルミスを含む）などがあった場合でも、その指定値は無視され処理が続行されることがあります。それにより期待する処理結果が得られないことがありますので注意して指定してください。

付録 A　エラーメッセージ

ここでは、出力されるエラーメッセージとその対処方法について説明します。

表 1：利用者画面で表示されるエラーメッセージ一覧

エラーメッセージ	対処方法
User ID or Password is incorrect. ユーザー ID またはパスワードが間違っています。 【原因】ユーザー ID またはパスワードが間違っています。	ユーザー ID とパスワードを確認しログインし直してください。
Failed to connect server. サーバへの接続でエラーが発生しました。 【原因】認証サーバと通信ができない状態です。	システム管理者に連絡してください。
Enter User ID and Password. ユーザー ID / パスワードを入力してください。 【原因】ログイン画面でユーザー ID またはパスワードが入力されませんでした。	ユーザー ID とパスワードを確認しログインし直してください。
Internal server error occurred. サーバでエラーが発生しました。 【原因】認証サーバでエラーが発生しました。	システム管理者に連絡してください。
Failed to get user data. ユーザーデータの取得でエラーが発生しました。 【原因】 プロキシエラーで認証サーバから情報が取得できませんでした。	
API Key is already present. API キーは既に作成されています。 【原因】複数ウィンドウ操作などで、すでに別ウィンドウで API Key が作成されているところに再度 API Key の作成を実行しました。	API Key はすでに作成されています。画面を再表示し最新の状態に更新してください。
API Key is not present. API キーは既に削除されています。 【原因】複数ウィンドウ操作などで、すでに別ウィンドウで API Key が削除されているところに再度 API Key の削除を実行しました。	API Key はすでに削除されています。画面を再表示し最新の状態に更新してください。
Parameter is invalid. 不正なパラメータが指定されました。 【原因】サーバとの通信時に不正なパラメーターが指定されました。	システム管理者に連絡してください。
An unexpected error occurred. 予期しないエラーが発生しました。 【原因】複数の要因でエラーが発生しました。	

用語集

D

Digital Annealer（デジタルアニーラ）
: 量子の振る舞いをデジタル回路で表現し、組合せ最適化問題を高速に解くハードウェアです。

F

FujitsuDA2MixedModeSolver
: Digital Annealer で QUBO を解くためのソルバーです。

FujitsuDA2PTSolver
: Digital Annealer でレプリカ交換機能を使用して QUBO を解くためのソルバーです。

FujitsuDA2Solver
: Digital Annealer で QUBO を解くためのソルバーです。

H

HOBO（Higher Order Binary Optimization）
: 組合せ最適化問題を高次単項式または高次多項式で表したものです。

Q

QUBO（Quadratic Unconstrained Binary Optimization）
: 組合せ最適化問題を二次多項式で表したものです。

W

Web API
: HTTP プロトコルを用いてネットワーク越しに呼び出す、アプリケーション間、システム間インターフェースです。

あ

アニーリング方式

金属工学における「焼なまし」を模したシミュレーション手法です。
金属の「焼なまし」では、高温に加熱して金属の原子が動きやすいようにしてからゆっくりと冷却することで、金属の結晶が安定した状態（＝エネルギーが低い状態）になり、金属内部の欠陥やひずみを取り除くことができます。
アニーリング方式によるシミュレーションは、金属の「焼なまし」と完全に同じ動作をするわけではありませんが、「焼なまし」と同じように、初めは「温度」に相当する擾乱（じょうらん）を決めるパラメーターを大きく取ることで状態が変化しやすい状況にしてから、「温度」を少しずつ下げていくことで最低エネルギー状態に近づけていく方法です。
与えられた初期状態からエネルギーが低い状態への変化は、乱数を用いて発生させた（温度により決まる）擾乱により確率的に起こります。このため、有限時間で必ず最低エネルギー状態が得られる保証はありませんが、時間をかけて温度を下げていくことで、最低エネルギー状態が得られる確率が時間とともに増加し、1 に近づいていきます。

さ

シミュレーテッドアニーリング

焼なましと呼ばれる過熱炉内の個体の冷却過程をシミュレートするアルゴリズムに端を発し、最適化問題（特に組合せ最適化問題）を解くための汎用近似解法の 1 つです。

ら

量子アニーリング

量子力学的な効果を利用して、組合せ最適化問題を解く手法です。

量子コンピュータ

量子力学的な「状態の重ね合わせ」を利用して、並列性を実現するとされるコンピュータです。

レプリカ交換

parallel tempering（並列焼戻し法）としても知られ、緩和が遅く、かつ局所解に陥りやすいシミュレーテッドアニーリング（焼なまし法）を改善する方法です。
方法としては、複数の初期化された異なる温度でアニーリングを並列に実行し、メトロポリス法（乱数発生により作った新しい状態を棄却するか採択するかの基準の与え方）の基準で、それぞれの温度間で状態を交換することで最適解を求めるものです。

FUJITSU Quantum-inspired Computing
Digital Annealer オンプレミスサービス ユーザーズガイド

発行日 2019 年 2 月