NEC Vector Annealing(x86版)
ユーザーズガイド
第2版

輸出する際の注意事項

本製品（ソフトウェアを含む）は、外国為替および外国貿易法で規定される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続きにあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

はしがき

- 本書は、x86システム上における、NEC Vector Annealing (VA) のインストール、環境設定及び、使用方法について説明したものです。

- 商標、著作権について
  - Linux はアメリカ合衆国及びその他の国におけるLinus Torvalds の商標です。
  - Red Hat、Red Hat Enterprise Linuxは米国およびその他の国において登録されたRed Hat, Inc.の商標です。
  - その他、記載されている会社名、製品名は、各社の登録商標または商標です。

用語定義・略語

用語・略語	説　明
アニーリング (Annealing)	ここではシミュレーテッドアニーリングを指し、徐々に「温度」を小さくすることで最適に近い解を確率的に求める手法になります。

用語	データ型	記述例	定義
スピン名	文字列	'x[0][0]', 'x[0][1]', …	各スピンの名前(必ずuniqueであること) ※スピン名に、'x'とb'x'のように文字列型とバイト列型を混在して指定した場合、別のスピンとして扱われるので注意してください。
スピンの状態	整数	0 or 1	各スピンの状態を0/1で記述する
スピン	[スピン名, スピンの状態]	['x[0][0]', 0], …	スピン名とスピンの状態の組み合わせで表したもの
収束エネルギー	実数*		モデル内のエネルギーの総和
制約条件の状態	ブール変数		制約条件を満たす=True, 満たさない=False
演算時間	実数*		アニーリングの計算時間

*:アニーリングエンジンは単精度浮動小数点のデータで受け付けます。
このため、Python APIで指定可能な実数のパラメータは単精度浮動小数点で表現可能な範囲(1.175494×10^{-38} から 3.402823×10^{38})で指定する必要があります。
上記の範囲外の値を指定した場合、infinityとして扱われ、エラーとなります。

動作要件

◆ 動作要件

■ x86 Linux

- Red Hat Enterprise Linux (RHEL) 9.2/9.4
- Rocky Linux 9.2/9.4
- RHEL 8.8/8.10
- Rocky Linux 8.8/8.10

■ 環境変数

- ノードライセンスをご利用の場合は、OMP_NUM_THREADSにサーバに搭載しているCPUのコア数を設定して下さい。
- 基本ライセンスをご利用の場合は利用可能なコア数とサーバに搭載しているCPUのコア数のうち、小さい値を設定してください。

■ Pythonバージョン

- Python 3.11

注意事項

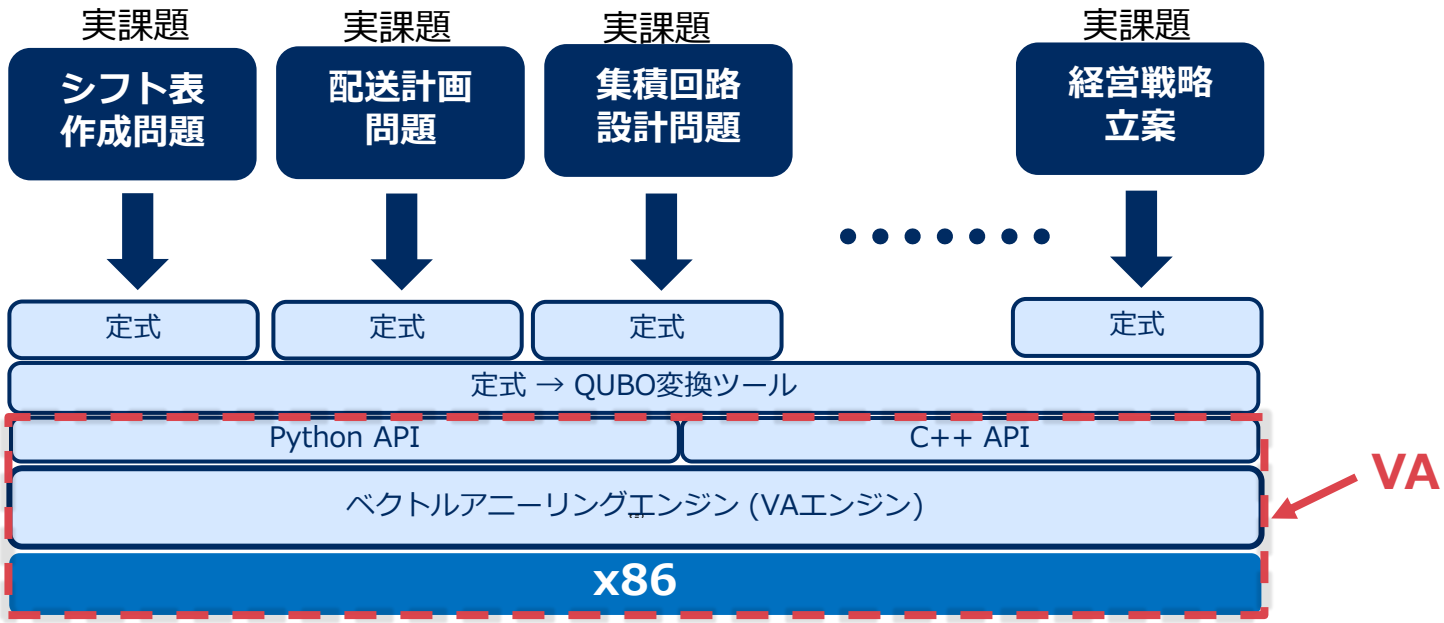
- 注意事項
  - 1CPUを複数のユーザで共有するなど、アニーリング処理が衝突する場合には実行時間が長くなってしまいます。そこで前の処理が終了するまで次の処理を投入しないなどのキュー制御を実装することを推奨します。
  - ハイパースレッディング環境においてはコアの割り当て状況によって性能が十分に発揮されない場合があります。性能を十分に発揮するためにはハイパースレッディング機能をOFFにして、実装されている物理コア数以下のスレッドを実行することを推奨します。
  - VAによって解くことのできる問題規模の上限は、マシンに実装されているメモリサイズによって変わります。

  - ログメッセージの冒頭に[va_qubo:OUT]もしくは[va_qubo:ERR]と出力されることがあります。
    [va_qubo:OUT]は、エンジンが標準出力にメッセージを出力していることを意味しています。
    [va_qubo:ERR]は、エンジンが標準エラー出力にメッセージを出力していることを意味しています。
  - warningメッセージにはユーザへの注意喚起だけでなく、サポートの際利用する情報が含まれており、ユーザ環境で発生した問題の原因を判断するのに利用することがあります。これにはユーザが定義したものではない情報も含まれていることがあります。

目次

NEC Vector Annealing (VA)とは

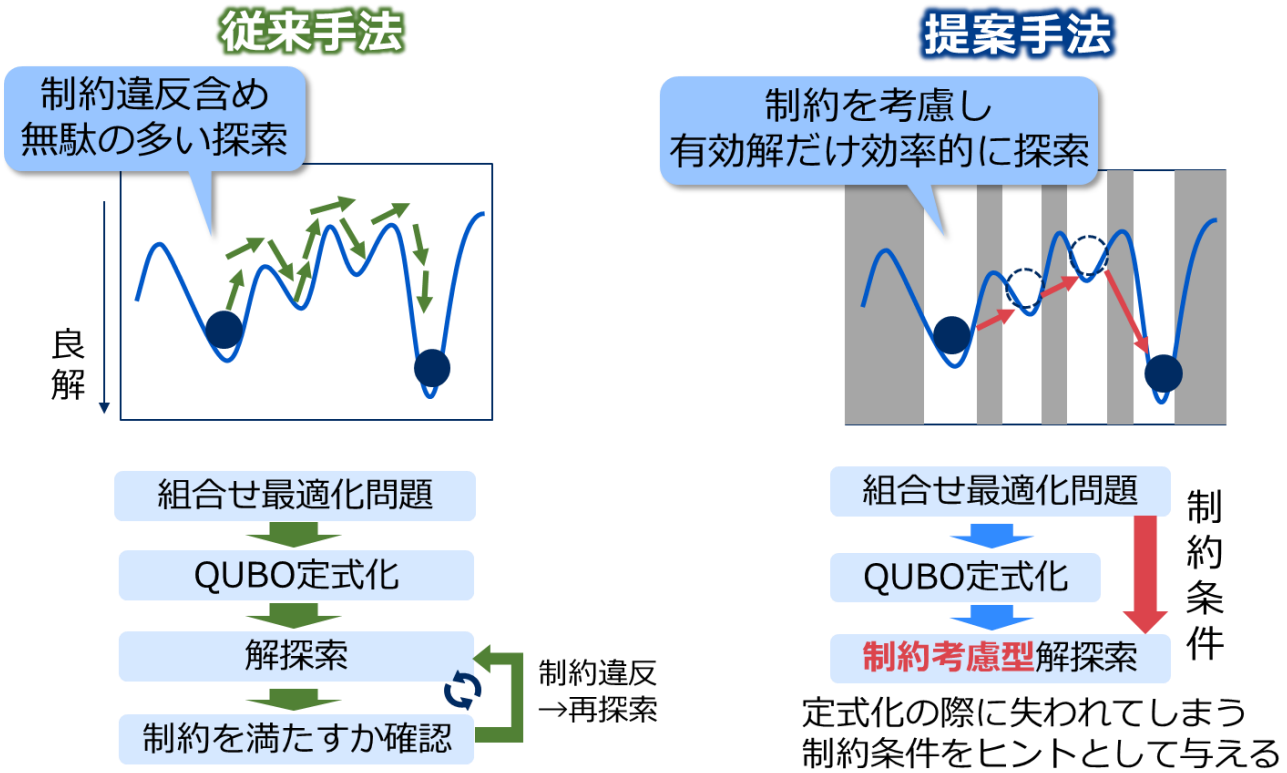
- x86上にシミュレーテッドアニーリングで求解する
  ベクトルアニーリング（VA）エンジンを実装し、組合せ最適化問題を
  定式化して解くソフトウェア
- スペック
  - 入力形式：QUBO形式（0/1のバイナリ）
  - 結合：解像度32bit階調の全結合
  - 出力形式：0/1のバイナリ列



VAの制約考慮型探索アルゴリズム

制約条件を満足する解空間のみを探索することで演算量を削減できる**フリップオプション**を用意

- 例：x1～x10のスピン(0/1)のうち1つだけ1になるという制約条件の場合、
  探索時にスピンが反転しても、上記制約は満たすようにx1～x10がセットで反転



VAご使用前の準備 (1/3)

- Python 3.11以上(推奨は3.11) 、numpy 2.0.0以上(推奨は2.0.0)、pyqubo* 1.4.0以上(推奨は1.4.0)が使えることが前提になります。

*：Pyquboとは：アニーリングマシンで組合せ最適化問題を解く際に、定式化した数式をQUBO形式に変換するドメイン固有言語（DSL）のことです。
Pythonの他のライブラリと同様に使用することができます。
https://pyqubo.readthedocs.io/en/latest/index.html
https://github.com/recruit-communications/pyqubo

- 1. 環境設定
  - 以下のコマンドを実行し環境変数を設定してください。 **（ログインの度に必要）**
    ログイン時に実行されるシェルの設定ファイルに設定を記載している場合は、省略可能です。
  - ノードライセンスをご利用の場合は、サーバに搭載されているCPUのコア数をOMP_NUM_THREADSに設定して下さい。
  - 基本ライセンスをご利用の場合はライセンスにより利用可能なコア数とサーバに搭載されているCPUのコア数のうち小さい値を設定してください。
    - 例えば、サーバに8コアのCPUが搭載されており、ライセンスにより8コア以上利用可能な場合は、8を設定してください。

```
$ export OMP_NUM_THREADS=8
```

- 2. 入手したVectorAnnealing-4.0.0X-4.0.0X-1.el9.x86_64.rpmを、root権限でyumでインストールして下さい。
  - VAエンジンはライセンスサービスに依存するため、 ライセンスサービスをVAエンジンより先か、または、VA エンジンと同時にインストールしてください。同時にインストールする手順は以下の通りです。

```
$ sudo yum install va_license_manager-4.0.0X-1.el9.x86_64.rpm VectorAnnealing-4.0.0X-4.0.0X-1.el9.x86_64.rpm
```

(注意）バージョンは変更される可能性があるため、この通りとは限りません。また、OSがRHEL 8.8/8.10またはRocky Linux 8.8/8.10の場合には、ファイル名の”el9”の部分が”el8”となります。

- 3. ユーザの環境変数の設定を確認し環境変数PYTHONPATHに以下のパスが設定されていない場合は、以下のパスを追加してください。

```
$ export PYTHONPATH=/opt/va/V4.0.0X/libexec/VectorAnnealing/python:${PYTHONPATH}
```

- 4. VA-ReleaseNotes-V4.0.0X.txtを開いて、”V4.0.0X”を確認してください。

```
$ cd /opt/va/V4.0.0X
$ cat VA-ReleaseNotes-V4.0.0X.txt
NEC Vector Annealing V4.0.0X Release Notes (November 2024)
```

VAご使用前の準備 (2/3)

5. /opt/va/V4.0.0X/samples/NP/python配下のサンプルプログラム(NP.py)を実行して、正常動作することを確認してください。

/tmpは書き込み権のある任意のフォルダ

```
$ cd /tmp
$ python3.11 /opt/va/V4.0.0X/samples/NP/python/NP.py

[HINTS]
 8 . . | . . . | . . .
 . . . 中略

[SOLUTION]
 8 1 2 | 7 5 3 | 6 4 9
 9 4 3 | 6 8 2 | 1 7 5
 6 7 5 | 4 9 1 | 2 8 3
 ------+-------+------
 1 5 4 | 2 3 7 | 8 9 6
 3 6 9 | 8 4 5 | 7 2 1
 2 8 7 | 1 6 9 | 5 3 4
 ------+-------+------
 5 2 1 | 9 7 4 | 3 6 8
 4 3 8 | 5 2 6 | 9 1 7
 7 9 6 | 3 1 8 | 4 5 2
```

左のようなSOLUTIONが出力されれば正常
※以下のようなWARNINGが時々出ることがありますが、求解できなかったためであり、再実行してみてください

```
2022-05-30 10:39:45.120 [Sampler] WARNING : va_qubo.exe
output result with broken constraint.
energy :   4.000 INVALID
```

6. 以上で準備完了です。

VAご使用前の準備 (3/3)

ご参考：他のサンプルコード（巡回セールスマン問題）

/opt/va/V4.0.0X/samples/TSP/python/TspLib.py

- 以下サイトからALL_tsp.tar.gzをダウンロードして解凍し、問題ファイル（例：eil51.tsp）を書き込み権のある任意のフォルダに格納してください。
  ※このうち、”EDGE_WEIGHT_TYPE：**EUC_2D**”の問題のみ対応（例：eil51.tsp）
  http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
  ※**100都市以上**のような大規模な問題ファイルはQUBOが大きくなり、Pyquboで生成できない可能性がありますので、ご使用しないようお願いいたします。
- 以下実行例のように実行できます。

```
$ cd /tmp
$ python3.11 /opt/va/V4.0.0X/samples/TSP/python/TspLib.py eil51.tsp
Read  eil51.tsp
Energy,  Length
 89.000, 437.000
---------------------------------------
[Result]
  Turn[] = [0, 21, 1, 28, 19, 35, 34, 2, 27, 30, 25, 7, 47, 22, 6, 42, 23, 13, 5, 26, 50, 45, 11,
46, 3, 17, 24, 12, 40, 18, 39, 41, 43, 16, 36, 14, 44, 32, 9, 38, 29, 33, 20, 15, 49, 8, 48, 4,
37, 10, 31]
  length = 437
```

書き込み権のある任意のフォルダ

ダウンロードしたTSPLIBのファイル

結果の距離

巡回する都市番号順の配列
(TSPLIBのnode番号-1が都市番号を表す)

VAモジュールの使い方 (1/2)

QUBOからアニーリングを実行

```
...

# quboを作成
param_C = max_len * 0.25
qubo, offset = model.to_qubo(feed_dict={'num_a': param_C})


# ---------------------------------------------
# Vector Annealing のモジュールをimport
import VectorAnnealing

# Vector Annealing のモデル作成
va_model = VectorAnnealing.model(qubo, offset)

# サンプラー作成
sa = VectorAnnealing.sampler()

# アニーリングを実行(実行回数に5回を指定)
result = sa.sample(va_model, num_reads=5)
```

アニーリング結果は、1回のアニーリング結果を以下のresultクラスの配列として返します。

```
-------------------------------------------
class result:
  def __init__(self, spin=None, energy=None, time=None, constraint=None, memory_usage=None):
    self.spin         = spin          # Spin state of QUBO. Ex) spin = { 'x':0, 'y':0, 'z':1 }
    self.energy       = energy        # Total energy of QUBO.
    self.time         = time          # Simulation time of this result.
    self.constraint   = constraint    # Satisfy constraint if constraint is 'True', Broken constraint if constraint is 'False'.
    self.memory_usage = memory_usage  # Memory usage of simulation.
-------------------------------------------
```

VAモジュールの使い方 (2/2)

PyquboでQUBOを作成

- 巡回セールスマン問題のケースより

```
# pyqubo を import
from pyqubo import Array,Placeholder

# quboを2次元で定義 : x[point_num][point_num]
x = Array.create('x', shape=(point_num, point_num), vartype='BINARY')

# 式を記述
param_a = Placeholder('num_a')
Hd = sum(sum((dlength[j][k] - d_min[j]) * x[i, j] * x[(i+1)%point_num, k] for i in range(point_num)) for (j,k) in pairs)
Ha = param_a * sum((sum(x[i, j] for i in range(point_num)) - 1)**2 for j in range(point_num))
Hb = param_a * sum((sum(x[i, j] for j in range(point_num)) - 1)**2 for i in range(point_num))
H = Hd + Ha + Hb

# モデルを作成
model = H.compile()

# quboを作成
param_C = max_len * 0.25
qubo, offset = model.to_qubo(feed_dict={'num_a': param_C})
```

パラメータの補足

Samplerクラスのパラメータ指定について

- アニーリングを実行する sa.sample() 関数は、num_reads以外にも、以下のパラメータをサポートしています

パラメータ名	type	default	説明
num_reads	整数型	1	サンプリングの回数
num_results	整数型	None	アニーリング結果の返却数 Noneもしくは1の場合、num_readsのうち 最良解のみを返す num_readsと同じ値の場合は、全結果を返す
num_sweeps	整数型	500	アニーリングのsweep数
num_threads	整数型	-1	アニーリングを実行するスレッド数 0以下もしくはNone(デフォルト)の場合、環境変数OMP_NUM_THREADSに指定した値で並列実行します。環境変数OMP_NUM_THREADSについては動作要件を参照ください。
beta_range	[実数型,実数型,整数型]	[10,100,200]	[start,end,nsteps]で温度の逆数であるβパラメータの開始(start)と終了(end)の値を指定。 nstepsは開始から終了までの分割数で省略可能。
init_spin	スピン配列	無し	初期スピン状態の指定 （詳細は後述の「初期スピン状態の指定 」の項を参照）
vector_mode	str型	VectorAnnealing.VECTOR_MODE_SPEED	VectorAnnealing.VECTOR_MODE_SPEED の場合、速度優先でアニーリング。 VectorAnnealing.VECTOR_MODE_CONSTRAINTの場合、制約優先でアニーリング。 VectorAnnealing.VECTOR_MODE_CONSTRAINTの場合、上記制約優先で制約を満たせば終了。
precision	整数型 (enumで定義した値で指定)	VectorAnnealing.PRECISION_COMPUTE_SINGLE	VectorAnnealing.PRECISION_COMPUTE_SINGLEの場合、単精度。 VectorAnnealing.PRECISION_COMPUTE_DOUBLEの場合、倍精度。

制約優先における注意事項

vector_modeのconstraintとconstraint_onlyについて

- フリップオプションを指定することにより、制約を満たすパターンを見つけます。
  （ただし、矛盾した設定のために制約を満足する解が無い場合は、無視されます）
  そのため、このモードを使用する場合はフリップオプションに関する定式を
  ハミルトニアンに含める必要はございません。
- 目的関数はなく、フリップオプションで指定する制約条件しかない問題では、
  constraint_onlyをご使用ください。
  制約条件を満たした時点で探索を終了します（アニーリングは行いません）。
  この場合、ハミルトニアンは不要になりますので、modelに対してquboとoffsetを以下のように入力し、spin_listにモデルで定義したスピンをリスト形式で指定してください。

```
qubo = {}
offset = 0
va_model = VectorAnnealing.model(qubo, offset, onehot=onehot_list, weighted_sum=weight_sum_list, spin_list=スピンのリスト)
sampler = VectorAnnealing.sampler()
```

補足）

- constraint_only：アニーリングしないため、目的関数などのハミルトニアンを入れなくても動きます
- constraint：目的関数などのハミルトニアンを入れないと動きません

- 本モードでしか使えない以下のフリップオプションがあります。
  - Pattern：禁止するスピンのパターンを制約に追加できます
  - Weighted sum：重み付き総和の値の条件を制約に追加できます
  - Conditioned min max one：min max one制約条件が有効となるスピンのパターンを制約に追加できます

3次以上の高次項における注意事項

QUBOでは2次項までしか扱えませんでしたが、3次以上の高次項もそのまま扱えるようになりました

- QUBOで3次項以上がハミルトニアンに出てきた場合、補助スピンと制約式を追加して2次項に変換してからアニーリングしていましたが、高次項のままアニーリングできるようになりました。
  高次項のスピン間のエネルギーはハミルトニアンではなく、以下のhigh_orderオプションで指定してください。

x[0][0]*x[0][1]*x[0][2]の係数は3.0

x[1][0]*x[1][1]*x[1][2]*x[1][3]の係数は4.0

```
high_order_list = {
    ('x[0][0]', 'x[0][1]',  'x[0][2]') : 3.0,
    ('x[1][0]', 'x[1][1]',  'x[1][2]',  'x[1][3]') : 4.0,
    ('x[2][0]', 'x[2][1]',  'x[2][2]',  'x[2][3]',  'x[2][4]') : -5.0
}

va_model = VectorAnnealing.model(qubo, offset, high_order=high_order_list)
```

複数のhigh_orderを定義

x[2][0]*x[2][1]*x[2][2]*x[2][3]*x[2][4]の係数は-5.0

- { ('x', 'y', 'z', 'z') : -1 } のような項の指定は、同じスピンの積はz*z = z の関係があるため、{ ('x', 'y', 'z') : -1 } と書く必要があります。なお3次項以上が設定可能です。

フリップオプションの使い方 (1/15)

フリップオプションの使い方

- フリップオプションを指定することにより、シミュレーションで効率よく解を求めることができます
  ※ただし、いずれのオプションもハミルトニアンの定式にも別途含める必要があります。
- 記述例(巡回セールスマン問題のケースより)

```
# create one hot constraint rule.
onehot = [0] * (2 * point_num)
for i in range(point_num):
    onehot1 = [0] * point_num
    onehot2 = [0] * point_num
    for j in range(point_num):
        onehot1[j] = 'x[%d][%d]' % (i, j)
        onehot2[j] = 'x[%d][%d]' % (j, i)
    onehot[2*i  ] = onehot1
    onehot[2*i+1] = onehot2

# create fixed spin constraint rule.
fixed = []
for i in range(point_num):
    for j in range(point_num):
        if (j == 0) and (i == 0):
            fixed.append(['x[0][0]', 1])
        elif (j != 0) and (i == 0):
            fixed.append(['x[0][%d]' % j, 0])
        elif (j == 0) and (i != 0):
            fixed.append(['x[%d][0]' % i, 0])

va_model = VectorAnnealing.model(qubo, offset, onehot=onehot, fixed=fixed)
```

Onehot 制約条件を定義

i番目には1つの都市のみ1となり、jの都市は1つの順番のみ1となるように制約を設定

j = 0の都市からスタートするようにx[0][0] = 1と固定し、それに伴いx[0][*] = x[*][0] = 0に設定

Fixed spin 制約条件を定義

Vector Annealing のモデル作成時に上記フリップオプションの制約条件を指定

フリップオプションの使い方 (2/15)

フリップオプションの記述方法

- One hot 制約条件
  - スピンのグループを定義し、指定したグループ内の1個のスピンのみが状態=1となる制約条件

One hot 制約条件のグループ

```
one_hot_list = [
  [ 'x[0][0]', 'x[0][1]',  'x[0][2]',  'x[0][3]',  'x[0][4]' ],
  [ 'x[1][0]', 'x[1][1]',  'x[1][2]',  'x[1][3]',  'x[1][4]' ],
  [ 'x[2][0]', 'x[2][1]',  'x[2][2]',  'x[2][3]',  'x[2][4]' ],
]

va_model = VectorAnnealing.model(qubo, offset, onehot=one_hot_list)
```

複数のone hot 制約条件を定義

onehot に、定義したone hot制約条件を指定

フリップオプションの使い方 (3/15)

フリップオプションの記述方法

- Fixed spin 制約条件
  ※このオプションのみ、ハミルトニアンの定式にも別途制約を含める必要はありません。
  - スピンの状態を指定した値(0/1)に固定する制約条件

スピン名とスピンの状態(0/1)の組で指定

```
fixed_spin_list = [
  [ 'x[0][0]', 1 ],
  [ 'x[0][1]', 0 ],
  [ 'x[0][2]', 0 ],
  [ 'x[0][3]', 0 ]
]

va_model = VectorAnnealing.model(qubo, offset, fixed=fixed_spin_list)
```

複数のスピンの状態を定義

スピンの値を格納した辞書型のデータをfixedに指定

フリップオプションの使い方 (4/15)

フリップオプションの記述方法

- And zero 制約条件
  - スピンのグループを定義し、指定したグループ内の少なくとも1個のスピンが状態=0となる制約条件

```
and_zero_list = [
  [ 'x[0][0]', 'x[0][1]',  'x[0][2]',  'x[0][3]',  'x[0][4]' ],
  [ 'x[1][0]', 'x[1][1]',  'x[1][2]',  'x[1][3]',  'x[1][4]' ],
  [ 'x[2][0]', 'x[2][1]',  'x[2][2]',  'x[2][3]',  'x[2][4]' ],
]

va_model = VectorAnnealing.model(qubo, offset, andzero=and_zero_list)
```

And zero 制約条件のグループ

複数のand zero 制約条件を定義

andzero に、定義したand zero制約条件を指定

フリップオプションの使い方 (5/15)

フリップオプションの記述方法

- Or one 制約条件
  - スピンのグループを定義し、指定したグループ内の少なくとも1個のスピンが状態=1となる制約条件

Or one 制約条件のグループ

```
or_one_list = [
  [ 'x[0][0]', 'x[0][1]',  'x[0][2]',  'x[0][3]',  'x[0][4]' ],
  [ 'x[1][0]', 'x[1][1]',  'x[1][2]',  'x[1][3]',  'x[1][4]' ],
  [ 'x[2][0]', 'x[2][1]',  'x[2][2]',  'x[2][3]',  'x[2][4]' ],
]

va_model = VectorAnnealing.model(qubo, offset, orone=or_one_list)
```

複数のor one 制約条件を定義

orone に、定義したor one制約条件を指定

フリップオプションの使い方 (6/15)

フリップオプションの記述方法

- Cubic supplement 制約条件
  - スピン x_1, x_2, y_1 が、必ず、式 $y_1 = x_1 x_2$ を満たす値をもつ制約条件
  - **high_orderオプションと同時に指定できません**

y_1, x_1, x_2 の順でcubic supplement 制約条件のスピンを記述

```
spl_list = [
  ['y[0]', 'x[0][0]',  'x[0][1]'],
  ['y[1]', 'x[1][0]',  'x[1][1]'],
  ['y[2]', 'x[2][0]',  'x[2][1]'],
]

va_model = VectorAnnealing.model(qubo, offset, spl=spl_list)
```

複数のcubic supplement 制約条件を定義

spl に、定義したcubic supplement制約条件を指定

※以下の例のような $y_1 = x_1 x_2$ を満たす制約式をハミルトニアンに追加する必要があります。

定式化例)

```
Hp1 = x[0,0]*x[0,1]-2*x[0,0]*y[0]-2*x[0,1]*y[0]+3*y[0]
Hp2 = x[1,0]*x[1,1]-2*x[1,0]*y[1]-2*x[1,1]*y[1]+3*y[1]
Hp3 = x[2,0]*x[2,1]-2*x[2,0]*y[2]-2*x[2,1]*y[2]+3*y[2]
```

フリップオプションの使い方 (7/15)

フリップオプションの記述方法

- Max one count 制約条件
  - スピンのグループを定義し、指定したグループ内の状態=1のスピン数が指定数以下となる制約条件

状態=1となるスピンの最大数を指定

Max one count制約条件を指定するスピンのグループを記述

```
max_one_list = [
  [ 2, [ 'x[0][0]', 'x[0][1]', 'x[0][2]', 'x[0][3]', 'x[0][4]' ] ],
  [ 1, [ 'x[1][0]', 'x[1][1]', 'x[1][2]', 'x[1][3]', 'x[1][4]' ] ],
  [ 1, [ 'x[2][0]', 'x[2][1]', 'x[2][2]', 'x[2][3]', 'x[2][4]' ] ],
]

va_model = VectorAnnealing.model(qubo, offset, maxone=max_one_list)
```

Max one count 制約条件の定義

複数のmax one count
制約条件を定義

maxone に、定義したmax one count制約条件を指定

フリップオプションの使い方 (8/15)

フリップオプションの記述方法

- Min max one 制約条件
  - 指定したスピンのグループに対して、状態が1となるスピン数の範囲を指定する制約条件
  - V4.0.0Xから記述フォーマットが変更されました。

  - Min max one 制約条件

```
constraint = [
  { 'min':2, 'max':4, 'spin_set':{ 'a', 'b', 'c', 'd', 'e', 'f' } },
  { 'min':2, 'max':4, 'spin_set':{ 'g', 'h', 'i', 'j', 'k', 'l' } }
]
model = VectorAnnealing.model(qubo, offset, minmaxone=constraint)
```

Min閾値の指定

Max閾値の指定

スピンの指定

spin_setに指定したスピンにおいて、状態が1となるスピンの個数をmin閾値以上、max閾値以下にする制約条件

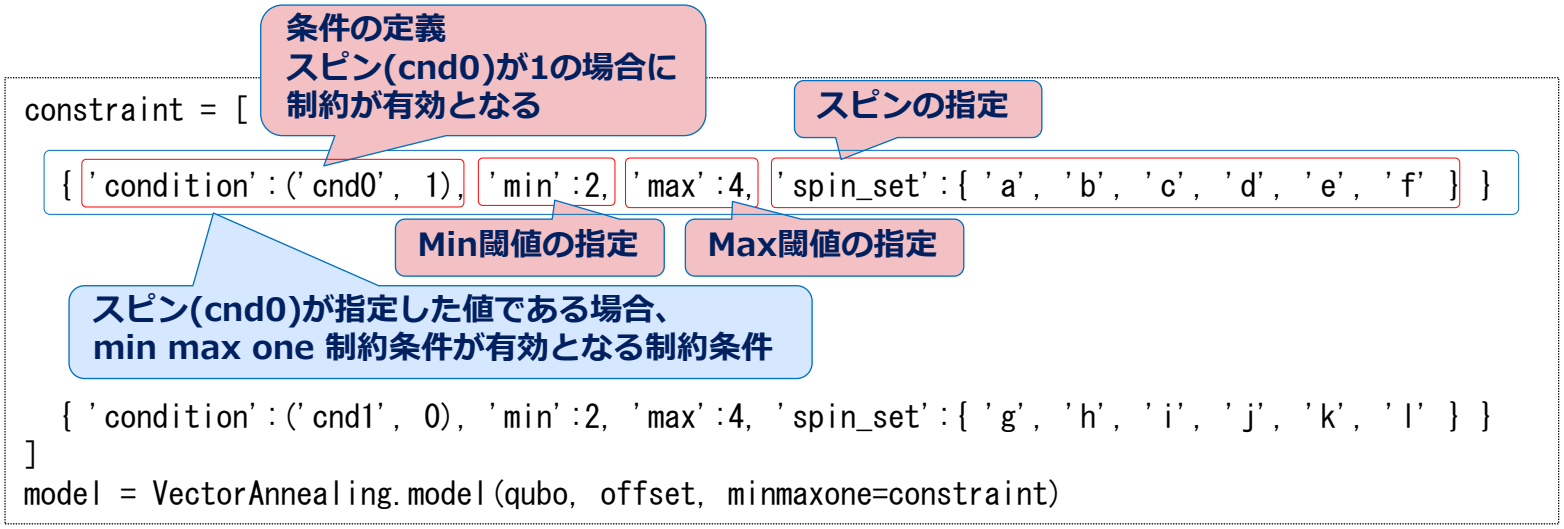
» min閾値 ≦ max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
min閾値 ≦ 状態が1のスピン数 ≦ max閾値

» min閾値 > max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
状態が1のスピン数 ≦ max閾値 もしくは min閾値 ≦ 状態が1のスピン数

フリップオプションの使い方 (9/15)

フリップオプションの記述方法

- Min max one 制約条件
  - Conditioned min max one 制約条件
    - 条件付きのmin max one制約条件はvector_modeがconstraintか、constraint_onlyの時のみ指定可能
    - ‘condition’に指定したスピンが指定した値である場合、min max one制約条件が有効となる制約
      指定した値でない場合は、min max one制約条件は無効となる




```
constraint = [
  {'condition':('cnd0', 1), 'min':2, 'max':4, 'spin_set':{'a', 'b', 'c', 'd', 'e', 'f'}}
  {'condition':('cnd1', 0), 'min':2, 'max':4, 'spin_set':{'g', 'h', 'i', 'j', 'k', 'l'}}
]
model = VectorAnnealing.model(qubo, offset, minmaxone=constraint)
```

» min閾値 ≦ max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
min閾値 ≦ 状態が1のスピン数 ≦ max閾値

» min閾値 > max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
状態が1のスピン数 ≦ max閾値 もしくは min閾値 ≦ 状態が1のスピン数

» min閾値=max閾値=1の場合
条件付きのOne hot制約条件として動作

フリップオプションの使い方 (10/15)

フリップオプションの記述方法

- Min max one 制約条件(旧フォーマット)
  - V4.0.0Xでは、旧フォーマットでのmin max one 制約条件の指定にも対応しています。
    - 各制約条件に対して最初に定義する値(左側)をmin、２つ目(右側)をmaxとします

スピンの指定

Min閾値

Max閾値

```
minmax_one_list = [
   [ 1, 3, [ 'x[0][0]', 'x[0][1]',  'x[0][2]',  'x[0][3]',  'x[0][4]' ] ],
   [ 2, 1, [ 'x[1][0]', 'x[1][1]',  'x[1][2]',  'x[1][3]',  'x[1][4]' ] ],
   [ 3, 2, [ 'x[2][0]', 'x[2][1]',  'x[2][2]',  'x[2][3]',  'x[2][4]' ] ],
]
va_model = VectorAnnealing.model(qubo, offset, minmaxone=minmax_one_list)
```

» min閾値 ≦ max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
min閾値 ≦ 状態が1のスピン数 ≦ max閾値

» min閾値 > max閾値の場合、制約条件に指定されたスピンが以下の状態を満たす
状態が1のスピン数 ≦ max閾値 もしくは min閾値 ≦ 状態が1のスピン数

フリップオプションの使い方 (11/15)

フリップオプションの記述方法

- Pattern 制約条件
  - Patterm制約条件はvector_modeが制約優先・制約限定モードの時のみ指定可能
  - 以下に示すスピンのパターンを制約条件として定義可能

  - Pattern 制約条件
    - パターン制約は実装されていません。FixedSpin制約条件を使用して記述してください

  - Conditioned pattern 制約条件
    - ‘condition’に指定したスピンが指定した値である場合、必ずスピンがpatternに指定した状態となる制約条件
      指定した値でない場合、patternに指定した制約条件は無効となる

条件の定義
スピン(cnd0)が1の場合に
制約が有効となる

スピンのパターンを定義
(a,b,c) = (0,0,0)

```
constraint = [
  { 'condition':('cnd0', 1), 'pattern':{ 'a':0, 'b':0, 'c':0 } },
  { 'condition':('cnd1', 0), 'pattern':{ 'd':1, 'e':0, 'f':1 } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint)
```

スピン(cnd0)が1の場合に、
(a,b,c) = (0,0,0) となる制約条件

フリップオプションの使い方 (12/15)

フリップオプションの記述方法

- Pattern 制約条件
  - Prohibited pattern 制約条件
    - –スピンがpatternに指定した状態以外の組み合わせとなる制約条件

```
constraint = [
  { 'prohibit':True, 'pattern':{ 'a':0, 'b':0, 'c':0 } },
  { 'prohibit':True, 'pattern':{ 'a':1, 'b':1, 'c':1 } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint)
```

‘pattern’が禁止パターンであることを指定

スピンのパターンの定義 (a,b,c) = (0,0,0)

2個の禁止パターン制約で、(a,b,c)が(0,0,0), (1,1,1)以外となる制約条件

  - Conditioned prohibited pattern 制約条件
    - –‘condition’に指定したスピンが指定した値である場合、prohibited pattern制約が有効となる
      指定した値でない場合、prohibited pattern制約が無効となる

```
constraint = [
  { 'condition':('cnd0', 1), 'prohibit':True, 'pattern':{ 'a':0, 'b':0, 'c':0 } },
  { 'condition':('cnd1', 0), 'prohibit':True, 'pattern':{ 'd':1, 'e':1, 'f':1 } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint)
```

条件の定義 スピン(cnd0)が1の場合に制約が有効となる

‘pattern’が禁止パターンであることを指定

スピンのパターンの定義 (a,b,c) = (0,0,0)

スピン(cnd0)が1の場合に、(a,b,c) が (0,0,0) 以外となる制約条件

フリップオプションの使い方 (13/15)

フリップオプションの記述方法

- Pattern 制約条件
  - Pattern equi spin 制約条件
    - スピンがPatternに指定した状態である場合、equi_spinに指定したスピンの状態が1となり、それ以外の場合は0となる制約条件

スピンがpatternに指定した状態を満たす場合、スピン(equi)が1となり、満たさない場合、スピン(equi)が0となる制約条件

```
constraint = [
  { 'equi_spin':'equi', 'pattern':{ 'a':0, 'b':0, 'c':0 } },
  { 'equi_spin':'cnd1', 'pattern':{ 'd':1, 'e':1, 'f':1 } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint)
```

    - Conditioned min max one 制約等の条件付き制約条件と組み合わせることで、スピンが特定のpatternを満たす時に、別の制約条件を有効とする記述が可能

スピンがpatternに指定した状態を満たす場合、スピン(cnd0)が1となり、conditioned min max oneが有効となるとなる制約条件

```
constraint1 = [
  { 'equi_spin':'cnd0', 'pattern':{ 'a':0, 'b':0, 'c':0 } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint1)

constraint2 = [
  { 'condition':('cnd0', 1), 'min':2, 'max':4, 'spin_set':{ 'g', 'h', 'i', 'j', 'k', 'l' } }
]
model = VectorAnnealing.model(qubo, offset, pattern=constraint1, minmaxone=constraint2)
```

フリップオプションの使い方 (14/15)

フリップオプションの記述方法

- Weighted sum 制約条件
  - vector_modeがconstraintか、constraint_onlyの時のみ指定可能。
  - スピンと重みのグループを定義し、指定したグループ内で、状態が1となるスピンの重みの総和範囲を指定する制約条件
    - 範囲を指定するパラメータとして、weighted_sum["comparison"]で定義してください
      - » 重み付き総和 ≦ 比較対象値である場合
        “comparison”:(VectorAnnealing.COMPARISON_OPERATOR_LESS_OR_EQUAL, 比較対象値)で指定
      - » 重み付き総和 = 比較対象値である場合
        “comparison”:(VectorAnnealing.COMPARISON_OPERATOR_EQUAL, 比較対象値)で指定
      - » 重み付き総和 ≧ 比較対象値である場合
        “comparison”:(VectorAnnealing.COMPARISON_OPERATOR_GREATER_OR_EQUAL, 比較対象値)で指定

スピン名1:係数1, スピン名2:係数2, … の重み付き総和

1*a + 2*b <= 1

1*c + 2*d = 2

1*e + 2*f >=2

```
constraint = [
    {"weight":{"a":1, "b":2}, "comparison":(VectorAnnealing.COMPARISON_OPERATOR_LESS_OR_EQUAL, 1)},
    {"weight":{"c":1, "d":2}, "comparison":(VectorAnnealing.COMPARISON_OPERATOR_EQUAL, 2)},
    {"weight":{"e":1, "f":2}, "comparison":(VectorAnnealing.COMPARISON_OPERATOR_GREATER_OR_EQUAL, 2)}
]
model = VectorAnnealing.model(qubo, 0, weighted_sum=constraint)
```

- 重みの係数と比較対象値は整数のみで実数は扱えません。ただし負の値は扱えます。

フリップオプションの使い方 (15/15)

フリップオプションの記述方法

- Weighted sum 制約条件
  - Conditioned weighted sum 制約条件
    - –‘condition’に指定したスピンが指定した値である場合、weighted sum制約が有効となる
      指定した値でない場合、weighted sum制約が無効となる

条件の定義
スピン(cnd0)が1の場合に
制約が有効となる

比較演算子、比較対象値

スピン名1:係数1, スピン名2:係数2,
… の重み付き総和

スピン(cnd0)が指定された値の場合に、
weighted sum制約条件が有効となる

```
constraint = [
    { 'condition':(cnd0, 1), 'weight':{'a':1, 'b':2}, 'comparison':(VectorAnnealing.COMPARISON_OPERATOR_LESS_OR_EQUAL, 1) },
    { 'condition':(cnd1, 0), 'weight':{'c':1, 'd':2}, 'comparison':(VectorAnnealing.COMPARISON_OPERATOR_EQUAL, 2) },
    { 'condition':(cnd2, 1), 'weight':{'e':1, 'f':2}, 'comparison':(VectorAnnealing.COMPARISON_OPERATOR_GREATER_OR_EQUAL, 2) }
]
model = VectorAnnealing.model(qubo, 0, weighted_sum=constraint)
```

初期スピン状態の指定

- 以下の2通りの方法で、アニーリング開始時のスピンの状態を指定することが可能です
  - モデルに対してFixedSpin制約条件を指定
    - Fixed Spin 制約条件は、スピンが固定であることを指定する制約条件であるため、指定したスピンの状態を初期スピン状態にも適用する
  - アニーリング実行時に初期スピン状態を指定

- 動作
  - アニーリング実行時に指定した初期スピン状態を初期値として使用します
  - 同一スピンに対して異なる値を指定した場合、以下の優先順位で値を設定してください
    1. Fixed spin 制約条件に指定した状態
    2. アニーリング実行時に指定した初期スピン状態

初期スピンの指定例）

```
spin_state = {
 'x[0][0]':0,
 'x[1][1]':1,
 'x[2][3]':1,
  …
}

sa = VectorAnnealing.sampler()
result = sa.sample(va_model, num_reads=5, init_spin=spin_state)
```

スピン名とスピンの状態(0,1)の組み合わせでスピン状態を指定
指定しないスピンは初期値なしとして扱い、アニーリング開始時に乱数で値を決定します

スピンの初期値を格納した辞書型のデータをinit_spinに指定

密行列モードと疎行列モード

Quboの非ゼロ要素数の密度により、密行列(dense)モードまたは疎行列(sparse)モードで内部処理を行います。

- 密行列モードと疎行列モードはsampler クラスのパラメータdenseで決定されます。

- dense (boolean)
  - Trueの場合：　密行列モード
  - Falseの場合：　疎行列モード
  - Noneの場合：　quboの非ゼロ要素数の密度が閾値より高い場合には密行列モード、低い場合には疎行列モードとします。密度の閾値は、環境変数VA_QUBO_DENSITY_THRESHOLDで変更することができ、0 ～ 1 の割合で指定してください。(default：0.03)

- モードの確認はVA_LOG_LEVEL=LOG_INFO_DETAILで出力したログで確認できます。

```
[va_qubo:OUT] [Weight matrix]
[va_qubo:OUT]   Mode              : dense
```

疎行列の速度改善モード

環境変数「VA_QUBO_FAST_SPARSE」の設定により、より高速に疎行列モードの実行が可能となります。

- 環境変数「VA_QUBO_FAST_SPARSE」をenableに設定(default: disable)

```
export VA_QUBO_FAST_SPARSE=enable
```

- enableに設定した場合、疎行列モードかつフリップオプションが設定されている場合に実行速度を高速化することができます。 一方で、enableに設定した場合は、高速化の可否にかかわらず疎行列モードのメモリ使用量が増加します

- disableに設定した場合、 従来通りの動作をします

V4.0.0X メモリ使用量削減における注意事項

V4.0.0Xでは、denseモードでのアニーリングの実行方式を変更し、メモリ使用量を削減しました。

- 実行モードの変更により、一部のログ出力が変更される場合があります。
- ログ出力の変更例については、Python API リファレンスガイドを参照してください。
- 従来の動作にしたい場合、環境変数VA_QUBO_RUN_ON_CHILD_PROCESS をenableに設定してください。(default: disable)

発行履歴

◆ 発行履歴一覧

2024年　1月　初版
2024年　11月　2版

◆ 追加・変更点詳細

初版　新規作成
2版　V4.0.0X向けに内容を修正

NEC Vector Annealing(x86版)
ユーザーズガイド

2024年　11月　 2版

日本電気株式会社
東京都港区芝五丁目７番１号
TEL(03)3454-1111（大代表）


Orchestrating a brighter world
NEC