量子アニーリングエミュレータのためのデータ構造

植田 圭[1,a)]　戸川 望[1]　木村 晋二[1]

概要：組み合わせ最適化問題の高速な解法として，近年量子アニーリングが注目を集めている．量子アニーリングでは，スピンとそれらの間の相互作用を考慮するイジングモデルを用い，最適化問題を系全体のエネルギーが最小となる場合に最適となるように表現する．量子アニーリングでは量子モンテカルロ法が用いられるが，演算回数が膨大なものとなり，エミュレータによる高速化が必要である．しかし，エミュレーションで使用できるメモリが限られているので，データ量を削減できるデータ構造の検討を行い，量子アニーリングの入力となる相関係数の行列を，値の連続性を用いて，疎行列となる場合に効率よく表す手法を提案する．具体的には，値の表にデータを二つずつ検索して登録する手法で，データ圧縮を行った．提案手法を 32 地点の巡回セールスマン問題に適用することで，データ量を約 1/10 に削減することができた．

キーワード：量子計算，イジングモデル，QUBO(Quadratic Unconstrained Binary Optimization)，巡回セールスマン問題，データ圧縮，疎行列の表現

Data Structure for Quantum Annealing Emulator

Kei Ueda[1,a)]　Nozomu Togawa[1]　Shinji Kimura[1]

Abstract: Recently, quantum annealing has attracted attention as an efficient algorithm for combinatorial optimization problems. In quantum annealing, an optimization problem is expressed by using the ising model, which is a set of correlated quantums and its total energy is minimum corresponding to the optimal solution. This manuscript introduces a compressed data structure to emulate the quantum annealing. To specify an ising model or an equivalent QUBO(Quadratic Unconstrained Binary Optimization) model, all coefficients between a pair of quantums and the self energy of each quantum need to be described. The data size becomes large and its reduction is important for the emulation since the usable memory size is limited. We propose a method to efficiently represent a sparse matrix of the correlation coefficients by using the continuity of the values. A value table are introduced and data are compressed by the search and registration method of two consecutive data in the value table. The proposed method is applied to input data of traveling salesman problems at 32 points and can reduce the amount of data to about 1/10.

Keywords: quantum computer, ising model, QUBO (Quadratic Unconstrained Binary Optimization), traveling salesman problem, data compression, sparse matrix representation

1. はじめに

半導体の微細加工技術にも限界が見え始め，計算機の性能向上にも限界が見えている．現状ではノイマン型コンピュータが広く用いられているが，組み合わせの大きい最適化問題では多大な計算時間を必要とし，従来とは異なるアルゴリズムや計算機構が必要とされている [1][2].

この問題を解決するために近年，従来方式とは異なる方式に基づく計算機の開発が活発化している．そこで注目されているのが，量子コンピュータであり，中でも量子アニーリングに基づく計算機構が注目されている [3]．量子アニーリングとは量子力学的に相関を持つ多数の量子ビットを制御し，同時に複数の状態をとる量子ビットが自ら，最もエネルギーの低い状態に変化する現象を利用することで，最適化問題を解く手法である．量子アニーリングは量

[1] 早稲田大学
Waseda University, 3-4-1, Okubo, Shinjuku, Tokyo, 169-8555 JAPAN
[a)] kei.ueda@islab.cs.waseda.ac.jp

子ビットを並列に扱うことで，類似手法のシミュレーテッドアニーリングに比べて，エネルギーの最小値を求める速度が指数関数的に速くなると言われている [4][5].

量子アニーリングでは，解を求めたい最適化問題を磁性体の振る舞いを説明するための統計力学上のモデルであるイジングモデルにマッピングする．そして，スピンの数，相関係数の数，スピン間の相関係数 J_{ij} と自己エネルギー h_i で表す．これらの入力に対し，量子アニーリングコンピュータはシステム全体のエネルギーを最小化するようにスピンを決定する．また，量子モンテカルロ法を用いて，量子アニーリングコンピュータの動作の模擬を行う手法も知られている．量子モンテカルロ法では，量子のスピンのトグルをランダムに行い，システム全体のエネルギーを最小化する．量子モンテカルロ法は通常の計算機でも実行できるが，非常に多くの繰り返し処理に大きな計算時間を必要とするため，エミュレータを用いた高速化が有効である．しかし，エミュレータでは，通常使用できるメモリの量が限られているため，スピン数の二乗にもなる入力データの削減が必要になる．相関係数は，0 要素を含む疎行列であることが多いので，非 0 要素を $(i,\ j,\ J_{ij})$ の 3 項組の並びで表すのが一般的である．これまでにも疎行列の表現法として Compressed Row Storage(CRS) 法が知られている [6][7]．これは相関係数の 3 項組を i でソーティングして，各 i の始まり番号を管理することで i を除いて $(j,\ J_{ij})$ の 2 項組で表すものである．これでデータの総量を削減できるが，十分とは言えない．

そこで，量子アニーリングのエミュレーションへ向けた入力データの圧縮法を提案する．提案手法は値の表を導入して，j が規則的に変化する場合に，値の並びのみで表すことでデータを圧縮する．とくに，値の表にデータを二つずつ検索して登録することで，値の並びのみで表せる箇所を見つける．本稿では提案手法を C 言語を用いて実現し，32 地点間の巡回セールスマン問題に適用しその評価を行う．

2. 量子アニーリングのエミュレーション

まず，準備として量子アニーリングを適用するうえで重要となるイジングモデルと QUBO (Quadratic Unconstrained Binary Optimization) 問題について述べ，その後に量子アニーリングについて述べる．

2.1 イジングモデルと QUBO

イジングモデルとは磁性を持つ量子の振る舞いを説明するための統計力学上のモデルである．s_i を i 番目の量子のスピンとする．スピンは ± 1 のいずれかの値をとる．+1 のときは上向きスピン，-1 のときは下向きスピンである．さらに s_i，s_j 間の相互作用係数を J_{ij}，スピン s_i 一体にかかる外部磁場を h_i と表す [8][9]．J_{ij} は s_i，s_j が相互作用していないときは 0 となる．スピン数を n とするとイジング

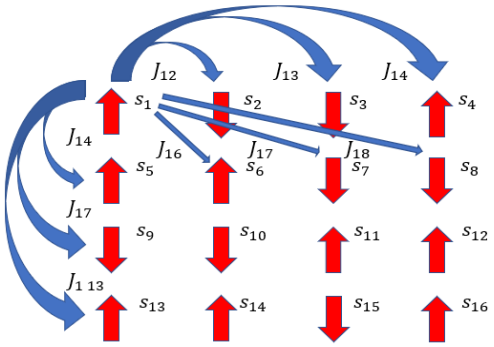

図 1 16 個のスピンの 2 次元配置の例.

モデルのエネルギー関数 H は，

$$H = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i}^{n} h_i s_i \qquad (1)$$

で定義される．実際の磁性体では，このエネルギー関数を最小化するようにスピンの向きが変化する．各種の最適化問題をこのイジングモデルで表すことで，実際の物理現象を用いて解くことができる．

16 個のスピンを 2 次元上に配置した場合の例を図 1 に示す．上向きが 1，下向きが-1 を表す．H の定義から，J_{ij} が正であると，s_i と s_j が同じ向きである場合にエネルギーが小さくなり．h_i が正である場合は s_i が上向きであるとエネルギーが小さくなる．

つぎに，QUBO とは Quadratic Unconstrained Binary Optimization の略で，1，0 を取る二値変数の値割当の最適化を考えるモデルである．QUBO は，変数の二次式で最適化のコストを表し，それが最小になるような変数への値割り当てを求める．種々の組み合わせ最適化問題を QUBO で表すことができる．また，QUBO をイジングモデルへ変換することも可能ある．QUBO の各変数 x_i をイジングモデルのスピン s_i に対応させ，以下の式に従って値を対応づけると，QUBO の最小解とイジングモデルのエネルギー最小の解が同じになる．

$$x_i = 0.5(s_i + 1) \qquad (2)$$

(2) 式から，イジングモデルのスピンが上向きで+1 をとるとき，QUBO の変数は+1 となる．同様に，イジングモデルのスピンが下向きで-1 をとるとき，QUBO の変数は 0 となる．イジングモデルの場合はスピンが 1 でも-1 でも加算しなければならないが，QUBO の場合は 1 の場合のみ加算すれば良いので演算回数を削減できる可能性がある．

2.2 量子モンテカルロ法を用いた量子アニーリングの模擬

量子アニーリングの模擬では，アドホックに決めた初期解から，系全体のエネルギーをコストとして，それが小さくなる方向に解の探索を行う．実際は量子モンテカルロ法

を用いてランダムにスピンをトグルさせる．ただし，単純な近傍探索法のようにエネルギーが H が小さくなる場合にのみスピンをトグルさせる方式では，局所最適解で探索が終了してしまう．そこで，量子アニーリングでは，スピンを変化させる過程で量子揺らぎを用いて局所最適解に落ち込むことを防いでいる．量子揺らぎが高いときには現在の解よりも悪くなる解（すなわちエネルギーが大きい解）の方向の探索も積極的に受け入れる．一方量子揺らぎが低い時には，コストが下がる方向にしか探索しないようにする．これにより，局所最適解に落ち込むことを避け，全域的な最適解を求める．具体的には，エネルギーが大きくなる場合でも遷移させる．β を温度の逆数，ΔH をエネルギー変化前後の差分として遷移確率は以下で表される．

$$\text{遷移確率} = \exp(-\beta \Delta H) \tag{3}$$

さらに，量子状態と呼ばれる複数の状態の重ね合わせを表すため，トロッタと呼ばれる複数のスピンの集合を扱う．異なるトロッタ間では，隣同士のトロッタ間でのみ温度に依存する相互係数に基づくエネルギー（横磁場に対応）を考える．系全体のエネルギーは以下のようになる [1].

$$\begin{aligned} H = \frac{1}{m}\sum_{k}^{m}(\sum_{i}^{n}\sum_{j}^{n} J_{ij} s_{i,k} s_{j,k} - \sum_{i}^{n} h_i s_{i,k}) \\ -\frac{1}{2\beta}\log\coth(\frac{\beta\Gamma}{m})\sum_{k}^{m}\sum_{i}^{n} s_{i,k} s_{i,k+1} \end{aligned} \tag{4}$$

m はトロッタ数を表す．トロッタは，複数の状態から探索を行うことに対応している．k はトロッタの識別変数を表し，β は温度の逆数，Γ は横磁場の強さを表す．Γ を徐々に下げていくことで，$\frac{\beta\Gamma}{m}$ の項が小さくなって干渉が高まり，最小解に収束する．

量子モンテカルロ法による模擬は以下のように行う．

(1) トロッタ数分の初期スピンをランダムに決定する．
(2) Γ を初期化する．
(3) ランダムにトロッタとスピンを選び，それをトグルさせた場合のエネルギーの差分を計算する．
(4) エネルギーの差分に基づく遷移確率の式 (3) でスピンをトグルさせるかどうか決める．具体的には 0〜1 の間の乱数を発生させてそれが遷移確率より小さければトグルさせる．
(5) 規定の回数だけ (3)(5) を繰り返す．
(6) Γ を 0.99 倍するなどして小さくする．
(7) (3)(7) を規定回数繰り返す．

上記からわかるように，量子アニーリングの模擬は二重のループから構成される．内側のループをインナーループ，外側のループをアウターループという．インナーループは 1 万回以上，アウターループは 1000 回以上としている．量子アニーリングは最適解を保証できないが一般的に精度の高い解が得られることが知られている．

3. 入力データの表現とその圧縮方法

ここでは，量子アニーリングの入力データの表現方法とその圧縮方法について述べる．

3.1 係数データの基本表現

量子アニーリングの入力は，最適化するエネルギーの計算に用いる係数 J_{ij} と h_i の値である．J_{ij} と J_{ji} は同じであるので，J_{ij} は $i < j$ の場合のみ示せばよい．最も単純には，J_{ij} と h_i を順番に列挙すれば良いが，スピンの数 n に対して J_{ij} は n の 2 乗個の半分，h_i は n 個となる．J_{ij} は 0 の要素が多い疎行列であることが多いため単純な列挙手法では効率が悪い．そこで J_{ij} については，非 0 要素を (i, j, J_{ij}) の 3 項組の並びで表す方法が用いられる．h_i についても，非 0 の h_i について (i, h_i) の 2 項組で表すことが考えられる．なおこの非 0 要素に着目した表現法は，0 要素が少ない場合は，インデックス部の情報の明示によりかえって記述の量が増えることに注意する．以下の巡回セールスマン問題を例として，入力データの表現方法を示す．

巡回セールスマン問題とは，N 個の都市とそれらの間の距離が与えられている時に，すべての都市を一度だけ回る経路の中で最短経路を求める問題である．経路の組み合わせの数は $N!$ であり，NP 困難問題のクラスに属することが知られている．

巡回セールス問題をイジングモデル（実際は QUBO）で表した場合の変数とそれらの間の係数について述べる．巡回セールスマン問題では，都市を回る順序を表すために，都市と都市数分の時刻を掛けた変数空間を考える．時刻 t に都市 a を通る場合に $x_{t,a}$ が 1，そうでない場合に 0 とする．

都市 a と b 間の距離 $d^{a,b}$ が与えられているとすると，全経路長は L は，

$$L = \sum_{t,a,b} d^{a,b} x_{t,a} x_{t+1,b} \tag{5}$$

と定義される．また，全ての都市を 1 回ずつ訪問する条件を満たすために，2 つの制約条件を追加する．

(1) 同時刻にセールスマンはどこか 1 都市にしかいない．
(2) 全ての都市は一度ずつしか訪問しない．

まず前者の制約（1）に関しては，

$$(\sum_{a} x_{t,a} - 1)^2 \tag{6}$$

と表せ，これが全ての t で成り立たなければならないため，

$$\sum_{t}(\sum_{a} x_{t,a} - 1)^2 \tag{7}$$

となる．制約（2）に関しても同様に，

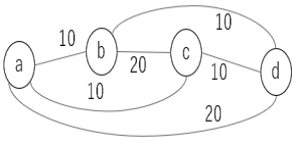

図 2 4 都市間の経路と距離.

```
16          1 6 20      2 7 10      4 8 200     7 8 20      10 11 200   -200
96          1 7 10      2 10 200    4 9 10      7 9 10      10 12 10    -200
0 1 200     1 9 200     2 12 10     4 10 10     7 10 10     10 13 20    -200
0 2 200     1 2 200     2 13 20     4 11 20     7 11 200    10 14 200   -200
0 3 200     1 3 200     2 14 200    4 12 200    7 15 200    10 15 10    -200
0 4 200     1 4 10      2 15 10     5 6 200     8 9 200     11 12 20    -200
0 5 10      1 5 200     3 4 20      5 7 200     8 10 200    11 13 10    -200
0 6 10      1 6 20      3 5 10      5 8 10      8 11 200    11 14 10    -200
0 7 20      1 7 10      3 6 10      5 9 200     8 12 200    11 15 200   -200
0 8 200     1 9 200     3 7 200     5 10 20     8 13 10     12 13 200   -200
0 12 200    1 12 10     3 11 200    5 11 10     8 14 10     12 14 200   -200
0 13 10     1 13 200    3 12 20     5 13 200    8 15 20     12 15 200   -200
0 14 10     1 14 20     3 13 10     6 7 200     9 10 200    13 14 200   -200
0 15 20     1 15 10     3 14 10     6 8 10      9 11 200    13 15 200   -200
1 2 200     2 3 200     3 15 200    6 9 20      9 12 10     14 15 200   -200
1 3 200     2 4 10      4 5 200     6 10 200    9 13 200                -200
1 4 10      2 5 20      4 6 200     6 11 10     9 14 20
1 5 200     2 6 200     4 7 200     6 14 200    9 15 10
```

図 3 4 都市の巡回セールスマン問題の入力データ.

$$\sum_{a}(\sum_{t} x_{t,a} - 1)^2 \tag{8}$$

と表せる．したがって，目的関数全体 H は，

$$H = \sum_{t,a,b} d^{a,b} x_{t,a} x_{t+1,b} + A\sum_{t}(\sum_{a} x_{t,a} - 1)^2 + A\sum_{a}(\sum_{t} x_{t,a} - 1)^2 \tag{9}$$

となる．A は正の定数であり，制約を満たすように適度に大きな値とする．この修正されたエネルギー関数 H をもとに係数を決める．まず (9) を展開し，$x_{t,a}^2 = x_{t,a}$ であることを用いて，自己エネルギー h_i は全てのスピンで $-2A$ となる．相互作用係数 J_{ij} は，自身とのコストを 0 とし，時刻 t+1 との距離 $d^{a,b}$ または $2A$ となる．

これらを踏まえ，図 2 に示す 4 都市の場合の QUBO の入力データの記述例を図 3 に示す．図 2 のグラフの枝のラベルが距離を表す．なお A としては 100 を用いる．入力として必要な情報は全部で 4 つある．1 つ目はスピンの数，2 つ目は 0 でない相互作用係数 J_{ij} の数である．3 つ目が相互作用係数 J_{ij}，4 つ目が自己エネルギー h_i である．この例では，4 都市であるので，スピン数は 16，J_{ij} の非ゼロ要素の数は 96，それから 96 個分の (i，j，J_{ij}) の並びとなる．その後，16 個分の h_i を並べて表す．J_{ij} と h_i は浮動小数点を用いるが，この例では整数値となっている．

3.2 疎行列のデータ圧縮

J_{ij} のような疎行列を表す手法としては，Compressed Row Storage(CRS) 法が知られている [6][7]．CRS ではインデックス部の記述量の削減を目的として，まず (i，j，J_{ij}) の並びを i でソーティングする．次に各 i について，何番目のデータから i に関連するかの情報を別途持つことで，i の情報を除いて (j，J_{ij}) の 2 項組の並びで表す．図 3 の例では各 i に対する先頭の情報 0，12，30，40，... と，(1，200)，(2，200)，...，(15，200)，(2，200)，(3，200)，(4，10)，... とで表せる．

3.3 値行列での連続性を用いた圧縮

節 3.2 では，疎行列のデータ圧縮方法を述べた．しかし，情報の削減だけでは十分ではないので，さらに削減する必要がある．そこで，値の表を導入して，入力データの値の連続性を用いて，j の情報も削減する手法を提案する．

CRS では，(j，J_{ij}) の 2 項組の並びで非ゼロ要素を表す．ところが図 3 からわかるように，(1，200)，(2，200)，(3，200)，(4，200) など同じ値が現れることがある．この場合，値表を別にして，そこへのインデックスで表すことで，値そのものを表すよりデータ量を削減できる．さらに j が順番に大きくなっているので，繰り返し回数を並記して，複数の (j，J_{ij}) を一度に表すことができる．例えば (1，4，200 へのインデックス) で 4 つ分のデータを表せる．0 は最初の j，4 は繰り返し回数を表す．なお，繰り返しの増分を導入して (1，4，+1，200 へのインデックス) とすることで，より一般的な繰り返しを表すことができる．

また，値の表を別にすることで (5，10)，(6，10)，(7，20) に対する値の列は，10，10，20 のように j の値とは独立に入れられるので，$i = 4$ に対する (9，10)，(10，10)，(11，20) に対してもそれをそのまま使うことができる．今回は，値表の方でもインデックスを変更する必要があるので，(5，3，+1，値の先頭のインデックス，+1) と表す．5 が最初の j，3 が繰り返し回数，+1 が j の増分である．最後の+1 は値表側のインデックスを増やすかどうかのフラグである．先の例は，(1，4，+1，200 へのインデックス，+0) となる．値表のインデックスの更新に関しては j のような一般化は行わず，+0 か+1 かの 2 通りとしている．

このような圧縮を行うため，(j，J_{ij}) の並びで J_{ij} が同じ場合をまとめることと，j が同じ差分で連続している場合を検出して，その場合は二つの J_{ij} をペアで表から検索・登録することとした．

図 3 に示した入力データを導入した記述法で表したものを図 4 に示す．s_j が 1 の時を例に挙げるとスピン s_1 を 4 回，差分 1 ずつとして繰り返す．値の表へのインデックスは 0 なので 200.00000 であり，+1 フラグが 0 なので同じ値で繰り返す．また，図 4 の 6 列目の整数のデータはスピン s_j の区切りとなる値で，スピン s_1 のデータが 5 つ並んだあとにスピン s_2 のデータが始めることを意味している．4 都市の場合，図 3 と図 4 を比較してわかるようにデータ量は削減できていない．CRS と同様，提案手法では元のデータ量が小さい場合には，かえって記述量が増える．

提案表現のデータ量は，(j の先頭，回数，増分，インデックス，フラグ) で，各々 (16bit，8bit，8bit，15bit，1bit) と

```
16          031 00    741 01    822 00    1331 00    0     0 15 12
80          441 71    1123 51   1131 01   021 51     5     13
141 00      1131 61   021 71    1421 71   131 61     10    200.000000
521 10      1421 51   221 51    222 01    523 01     14    10.000000
721 21      021 01    431 00    521 21    931 61     18    20.000000
1221 01     221 11    821 71    721 51    1222 00    23    200.000000
1421 11     541 00    1021 51   922 00    1511 00    28    10.000000
022 00      921 10    1511 00   1241 11   031 11     32    10.000000
331 01      1121 21   024 00    331 61    622 01     38    200.000000
621 71      041 51    521 10    621 51    921 21     44    20.000000
923 01      422 00    722 21    831 00    1121 51    49    10.000000
1331 61     731 01    1031 00   1221 71   1322 00    54    10.000000
021 00      1021 71   1321 10   1421 51   021 71     59    200.000000
341 01      1311 00   1511 20   021 01    221 51     64    200.000000
723 51      041 11    123 01    231 11    731 61     70    -200.000000
1241 11     421 00    531 61    821 01    1021 51    75
                                1021 11   1231 00    81
                                                     1
```

図 4 4 都市の TSP における圧縮後の入力データ．

すると，48bit となる．一方，$(j,\ J_{ij})$ の方は (16bit，32bit) なので，同じ値が続く場合は 2 つ以上，値の連続性の場合 j は 3 つ以上で提案手法が有利となる．なお，h_i については値のみの並びで表されているが，$(i,\ h_i)$ と考えて CRS を適用するとともに，値の繰り返しを導入する．

4. 実現と評価

ここでは，実際の圧縮法を述べ，32 地点間の巡回セールスマン問題に適用したときの評価を行う．

4.1 圧縮法の実現

圧縮は，イジングモデル（QUBO）に基づく入力に対して行われる．$(i,\ j,\ J_{ij})(i<j)$ の入力に対し，まず $i>j$ に対しても $(i,\ j,\ J_{ij})$ が出るように重複させ，i と j でソーティングしておき，それに対して圧縮を行い，さらにそれを定数行列として.h ファイルに変換し，量子アニーリングエミュレータ（シミュレータ）に渡す．$(i,\ j,\ J_{ij})$ を $(i>j)$ に対して重複させるのは，量子アニーリングのスピンのトグル時の処理の効率を考えてのことである．なお，都市間の距離の表からイジングモデルの記述を作成するプログラムも作成した．

その入力を圧縮する過程を図 5 に示す．入力から圧縮を行い，最適解を求めるまでに input，duplicate，compress，.h ファイルの作成の 4 つのプログラムを C 言語を用いて実現した．まず，入力をイジングモデルに基づき決定する．これは，スピンの数，相互作用係数の数，相互作用係数 J_{ij}，自己エネルギー h_i の 4 つの情報である．先の図 3 が具体例である．つぎの duplicate 部分では，これまでスピン s_i において，$i<j$ の場合のみを記述していたが，$i>j$ も明示して持つ．あるスピンをトグルさせた場合のエネルギーの差分計算時に，i に関連する全ての J_{ij} が必要なためである．リンクを用いることで，$i<j$ のみの情報で $i>j$ の情報もたどることもできるが，リンク情報が大きくなるため，情報を duplicate するのが良いと判断した．なお，duplicate でデータサイズは 2 倍となる．duplicate したデータは i と j でソーティングされたものとなっている．compress では

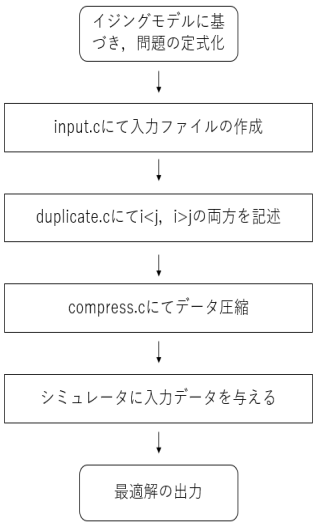

図 5 入力圧縮のフローチャート．

```
spin 16 //スピン数
numh 1 //h の数
numva 13 //値表の数
Read the ‘compress’ file
result
1 0 0 0
0 1 0 0
0 0 0 1
0 0 1 0
```

図 6 4 都市の巡回経路の出力．

表 1 シミュレータの初期パラメータ．

項目	値
初期解	ランダム
β	100.0
Γ	1.0
トロッタ数	30
Inner Loop	10000
Outer Loop	1000

3.3 で述べたデータ圧縮を行う．図 4 が圧縮後の例である．

また，圧縮後のデータを入力として動作する量子アニーリングシミュレータも作成した．表 1 に本シミュレータが用いているパラメータを示す．本シミュレータは C 言語で記述され，乱数を用いて量子モンテカルロ法を実現している．本シミュレータでは，入力を定数配列の形で定義し，コンパイルして実行する．4 都市の巡回セールスマン問題における最適解の出力例を図 6 に示す．今回の例題では a → b → d → c が最短経路となる．量子数 1024 の 32 地点の巡回セールスマン問題に対しても動作を確認している．

4.2 圧縮法の巡回セールスマン問題での評価

実験環境は，Intel Core i5-8400CPU@2.80GHz で 8.0GB 主記憶の計算機で Linux OS を用いて行った．今回は，32 地点の巡回セールスマン問題の距離のみを変更した 3 つの入力データを作成し，それに対して圧縮を行った．まず圧縮前後のファイルのバイト数での評価を表 2 に示す．圧縮前は，$i<j$ の場合のみの $(i,\ j,\ J_{ij})$ 形式のファイルのサイズ（バイト数）である．また，圧縮後は duplicate した後で圧縮を行った結果のファイルのサイズを表す．example1 と example2 は 32 地点の都市の座標をランダムに決めた．example3 は 32 地点の都市の A，B，C，... の座標が (1,1)，

表 2　圧縮結果.

	圧縮前 [Byte]	圧縮後 [Byte]	圧縮率 [%]
example1	1,226,468	137,612	11.220
example2	1,232,484	127,310	10.330
example3	1,227,108	414,594	33.786

表 3　ビット幅を考慮した圧縮結果.

	圧縮前 [bit]	圧縮後 [bit]	圧縮率 [%]
example1	4,096,000	368,528	8.997
example2	4,096,000	341,600	8.340
example3	4,096,000	1,084,960	26.488

(2,2)，(3,3)，... と規則的になるように設定した．結果として，example1 と example2 ではデータ量が約 1/10 に削減され，example3 ではデータ量が約 1/3 となった．圧縮に必要な計算時間はいずれも 1 秒以下であった．

つぎに，エミュレーション内部での，各データのビット幅を考慮したサイズの比較を行う．(i，j，J_{ij}) では 16bit，16bit，32bit の 64bit に J_{ij} の個数を掛けたものと，h_i に対する 32bit にスピン数を掛けたものである．一方，提案手法では，各スピンに対して J_{ij} の表の先頭 (16bit)，J_{ij} の表のデータとして，j の初期値 (16bit)，繰り返し回数 (8bit)，差分 (8bit)，値表へのインデックス (16bit)，フラグ (1bit，ただしインデックスの 16bit に組み込むので 0) の 48bit に表のサイズを掛けたもの，値の表として値の浮動小数点数の 32bit に表のサイズを掛けたものとなる．example1 では圧縮前は 63488*64+1024*32bit となる．圧縮後は，1024*16+6549*48+1181*32 である．example2 でも圧縮前は 63488*64+1024*32 であり，圧縮で 1024*16+5984*48+1187*32 となる．example3 でも，圧縮前は 63488*64+1024*32 であり，1024*16+21488*48+1161*32 となる．結果を表 3 に示す．

example3 は都市を最初から順番にたどる場合に最適解となる．また距離も $s_i \to s_{i+1}$ を d としたときに，$s_i \to s_{i+2}$ は $2d$ となっている．圧縮率がこの問題で悪くなった理由は不明であり，今後解析を進めるとともに，ヒューリスティックな圧縮手法の改良を行う．また，問題によって適切なパラメータ β，Γ，トロッタ数を解析的に求める手法を検討する．

5. おわりに

本稿では，量子アニーリングを実現する汎用エミュレータのためのデータ表現方法について検討を行い，エミュレータにおけるメモリの制限を考慮し，入力データの圧縮法を提案した．この手法は，疎行列の表現で，値の表を導入し，連続する二つの値をペアで検索・登録することで，データ圧縮を行う．例として扱った巡回セールスマン問題での相関係数となる行列に値の連続性かつ疎行列という特徴を捉え，効率よく圧縮することができた．32 地点間の巡回セールスマン問題に適用した場合，複数の問題で 1/10 程度への削減を確認できた．巡回セールスマン問題以外でも，スロット配置問題など同様の性質を持つ疎行列に対しては圧縮が可能であると考えられる．

今後は，別の最適化問題に対しても適用し，表現方法の有効性を検討する予定である．また，高位合成を用いた FPGA でのハードウェア生成と，実際にエミュレーションした場合の評価と，エミュレーション時のパラメータ設定方法も明確にすることも今後の課題である．

謝辞　早稲田大学・情報システム研究室の柳澤政生教授，史又華教授，吉増敏彦教授はじめメンバーの皆様には日頃からのご討議を心から感謝します．また，早稲田大学戸川研究室の田中宗准教授他メンバーの皆様にも日頃からのご討議に感謝します．この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）委託業務の結果，得られたものである．また，NEDO のプロジェクトを総括いただく、NEC のご担当の皆様に感謝します．

参考文献

[1] Takuya Okuyama, Masato Hayashi, and Masanao Yamaoka “An Ising computer based on simulated quantum annealing by path integral Monte Carlo method,” *Proc. of IEEE International Conference on Rebooting Computing*, pp.1-6, November 2017.

[2] Jasmeet Singh and Mohit Singh“Evolution in Quantum Computing,” *Proc of IEEE International Conference System Modeling & Advancement in Research Trends*, pp.1-4, November 2016.

[3] 塚本 三六, 高津 求, 松原 聡, 田村 泰孝“組み合わせ最適化問題向けハードウェアの高速化アーキテクチャー,” *Fujitsu Vol.68*, pp.8-14, June 2017.

[4] Elizabeth Crosson, Aram W. Harrow“Simulated Quantum Annealing Can Be Exponentially Faster than Classcal Simulated Annealing,” *Proc. of IEEE 57th Annual Symposium on Foundations of Computer Science*, pp.1-10, October 2016.

[5] Pierre Berg，Baptiste Cavarec，Arpad Rimmel，Joanna Tomasik “Restricting the search space to boost Quantum Annealing perfoemance,” *Proc. of IEEE Congress on Evolutionary Computation*, July 2016.

[6] 曽我 尚人，佐藤 真平，中原 啓貴 “Sparse Robust Autoencoder による心電図外れ値検出器のハードウェア向けのモデル圧縮について，” 信学技報 *IEICE Technical Report VLD2018-114*，February 2019.

[7] Ryan Kastner, Janarbek Matai, and Stephen Neuendorffer “Parallel programming for fpgas.” *CoRR*, Vol. abs/1805.03648, 2018.

[8] 金丸 翔, 於久 太祐, 多和田 雅師, 田中 宗, 林 真 人, 山岡 雅直, 柳澤 政生, 戸川 望, “イジング計算機によるスロット配置問題の解法, ” 信学技法　*VLD2018-34*, Vol. 118, No. 83, pp.161-166, June 2018.

[9] Masanao Yamaoka, Chihiro Yoshimura, Masato Hayashi, Takuya Okuyama, Hidetaka Aoki, and Hiroyuki Mizuno “A 20k-Spin Ising Chip to Solve Combinatorial Optimization Problems With CMOS Annealing,” *Proc. of IEEE JOURNAL OF SOLID-STATE CIRCUITS*, VOL. 51, NO. 1, January 2016.