Compressed Input Data Format of Quantum Annealing Emulator

Sohei SHIMOMAI*, Kei UEDA+, Shinji KIMURA*
(*Waseda University, +Fujitsu)

Proc. of DCC 2023, p.362.

Outline

- Introduction
- Ising Model
- Annealing
- Traveling Salesman Problem
- Input Data Format
  - Conventional Data Format
  - Existing Compression Method: CSR
  - Proposed Method
- Evaluation
- Conclusion

Introduction

Data compression is important for SQA emulator

- Quantum Annealing (QA)
  - Solve combinatorial optimization problems by mapping to Ising model
  - Utilize quantum phenomena
  - Physical spin can generate at near absolute zero temperature
- Simulated Quantum Annealing (SQA)
  - Simulate QA on conventional digital computer/circuits
- SQA > QA {make easy, less price, variable, scalable}
- The input data：Interact coefficient $J_{i,j}$, The magnetic fieldh_i
  - Number of spin: N J_{ij} →$N \times N$ array h_i→N

Note that memory size and bandwidth are limited on hardware emulation

Compression is important since the size is N^2

Contribution of This Work

Propose a new data format for input sparse matrix

◆Proposed method

- Focus on repetition and value sequence in indexes of Coordinate representation
- Prepare a list of value itself and refer to it by index

→ Reduction of data size

◆Evaluation

- Apply to the traveling salesman problem
  - This input data has high repetition and value sequence
- Implement simulator/emulator using compressed data directly

Ising Model

Annealing can solve combinatorial optimization problem quickly

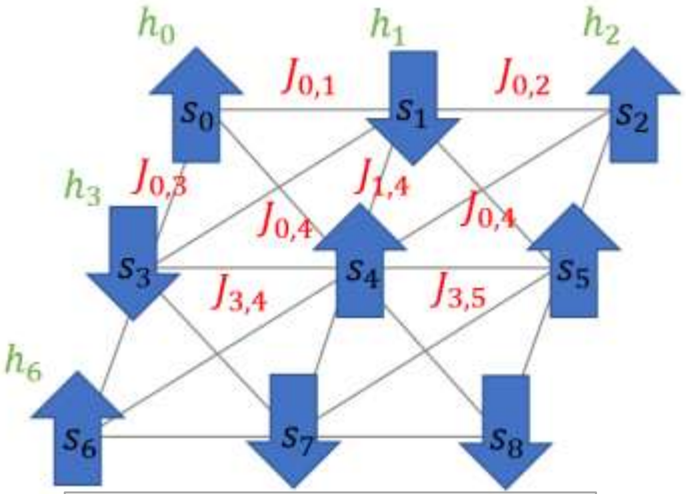
Ising Model

- Describe the spin behavior of magnetic materials in statistical mechanics
- Physical spins decides their direction to minimize total energy
- Map a combinatorial optimization problem to Ising model

Quantum Annealing (QA)

- Method to obtain the minimum energy state
- Magnetic field is applied and gradually reduced
- Similar to Simulated Annealing but uses quantum effects
- Simulation method has been proposed by toggling spins randomly

→ Search for globally optimal solution

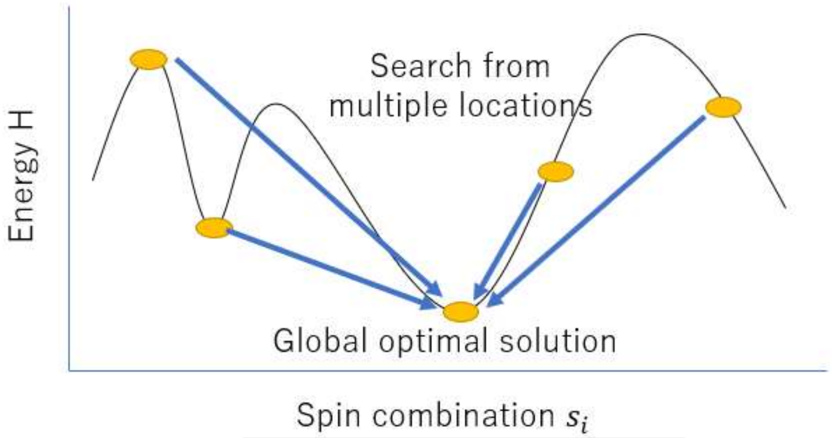


2-dimension Ising model.

Energy function

$$H = -\sum_{i<j} J_{i,j}\, s_i s_j - \sum_i h_i s_i$$

Spin	$s_i \in \{-1, 1\}$
Interact coefficient	J_{ij}
Self energy	h_i



Quantum Annealing.

Simulated Quantum Annealing (SQA)

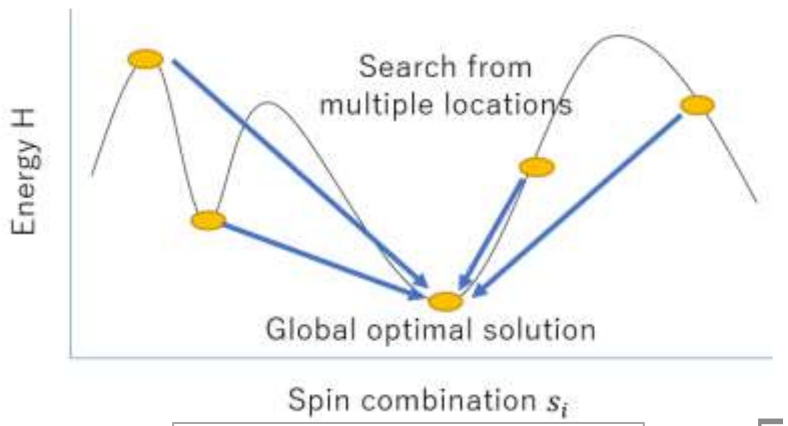
SQA method utilizing quantum superposition

- Multiple copies of spin set (called trotter) are used for quantum effect
- Trotters interfere with each other to search for the optimal solution
- $s_{i,k}$ is the i-th spin of the k-th trotter
- Add a transverse magnetic field term to represent quantum superposition

$$H = \frac{1}{m}\sum_{k=1}^{m}\left(-\sum_{i<j} J_{ij}s_{i,k}s_{j,k} - \sum_{i=1}^{n} h_i s_{i,k}\right) - \frac{1}{2\beta} logcoth\left(\frac{\beta\Gamma}{m}\right)\sum_{k=1}^{m}\sum_{i=1}^{n} s_{i,k}s_{i,k+1}$$

(J_{ij} : Interact coefficient, h_i : Magnetic field, m : Trotter’s ID,
k : Trotta‘s identification number, β : Inverse of temperature,
Γ : transverse magnetic field coefficient)

- The transverse magnetic field is gradually decreasing



SQA transition image.

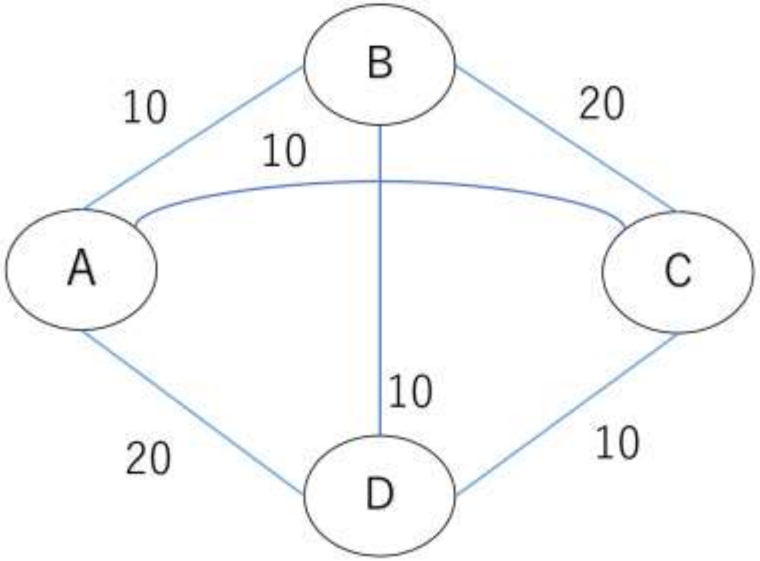
Traveling Salesman Problem（TSP）

Well known typical combinatorial optimization problem

- The problem of finding the route with the shortest total distance among routes that visit all cities only once
- Example: One of the shortest pass **B→A→C→D**
- $x_{t,a}$ is binary variable taking 1 when a traveler passes a city "a" at time "t"
- Constraints

(1)Typical Combinatorial Optimization Problem, (2) Each city is visited only once

→(1)$\sum_t x_{t,a} = 1$ for all a, (2) $\sum_a x_{t,a} = 1$ for all t



Example of 4 city TSP.

Energy function H:

$$H = \sum_{t,a,b} d_{a,b} x_{t,a} x_{t+1,b} + A \sum_t (\sum_a x_{t,a} - 1)^2 + A \sum_a (\sum_t x_{t,a} - 1)^2$$

($x_i \in \{0,1\}$, $d_{a,b}$:Distance between a and b, A :Penalty)

Conventional Data format

The input data consists $J_{i,j}$ and h_i

- **Input data**: $n \times n$ matrix of $J_{i,j}$ and h_i
  - Upper triangular matrix is required
- **Array format**:
  - $n \times n$ upper triangular matrix of $J_{i,j}$ and h
- **COO (Coordinate) format**:
  - Write only nonzero J_{ij}
  - As a list of $(i, j, J_{i,j})$ when 0 elements are many h_i is represented in a similar manner
- Use properly by nonzero density and bit width

```
0, 80, 70, 10, 10
    0, 40, 30,  0
        0, 80, 70
            0,  0
                0
```

Array

```
1, 2, 80
1, 3, 70
1, 4, 10
1, 5, 10
2, 3, 40
2, 4, 30
3, 4, 80
3, 5, 70
```

COO

$J_{i,j}$ of Array and COO format.

> In this research,
> focus on COO format

Existing Compression Method: CSR(Compressed Sparse Row)

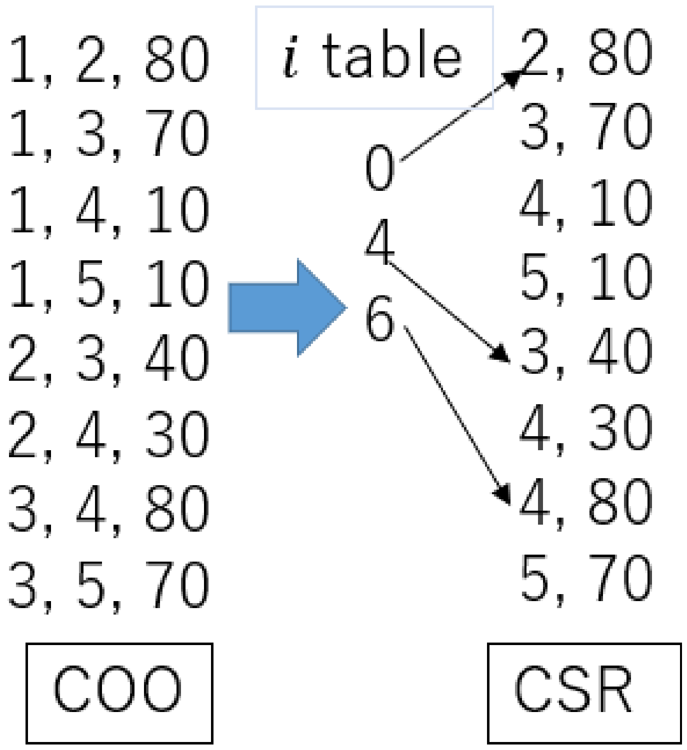
Reduce i-element description

◆CSR(Compressed Sparse Row)[1]

- First, in (i, j, J_{ij}), sort by i
- (i, j, J_{ij}) in COO

→$(j,\ J_{ij})$ + Number of first lines of each i + h_i

- Write number of first lines of each i
  - If want refer J_{ij} of $i = 0$ → Refer (j, J_{ij}) in the lines from 0 to 3



Compress from COO to CSR.

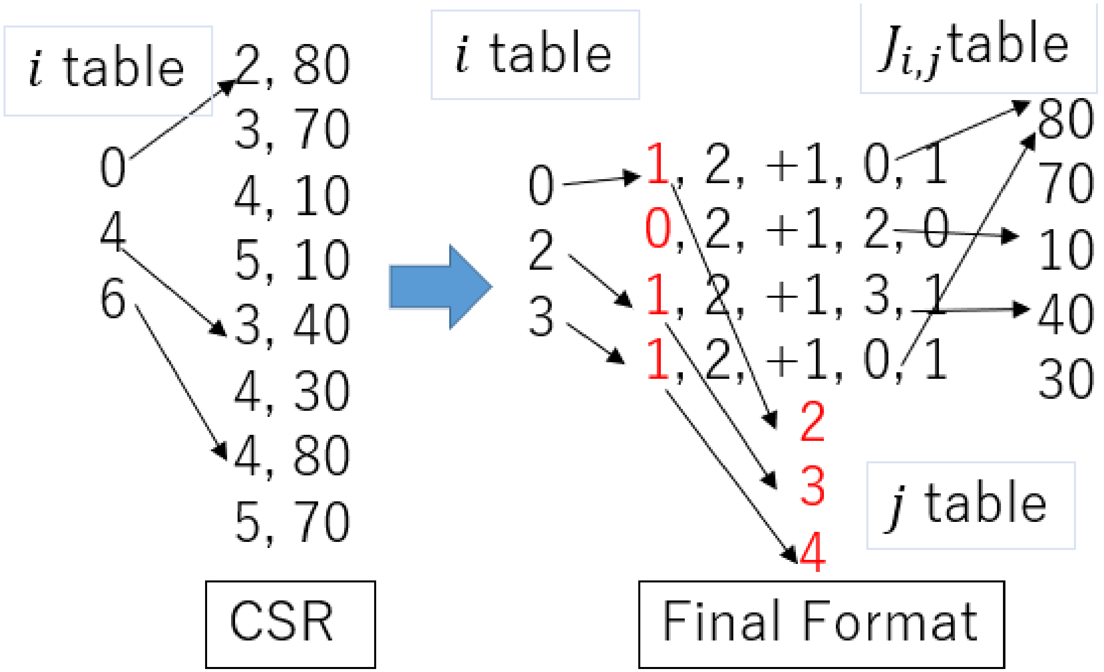
[1] Aydin Buluc, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson, Parallel sparse matrix-vector and matrix-transpose vector multiplication using compressed sparse blocks, SPAA '09, pp. 233-244, Aug. 2009.

Proposed Method

j information in the list $(j, J_{i,j})$ of CSR has been compressed

- From COO format input data, based on CSR, compress by rewriting the data format with focus on
  - ① Repetition of J_{ij}
  - ② Value sequence of $J_{i,j}$
  - ③ Continuity of j
- Write J_{ij} as 5 elements of index and Using dictionary and index

Purpose:
To describes mainly indexes,
- Reduction of description
- Reduction of bit width

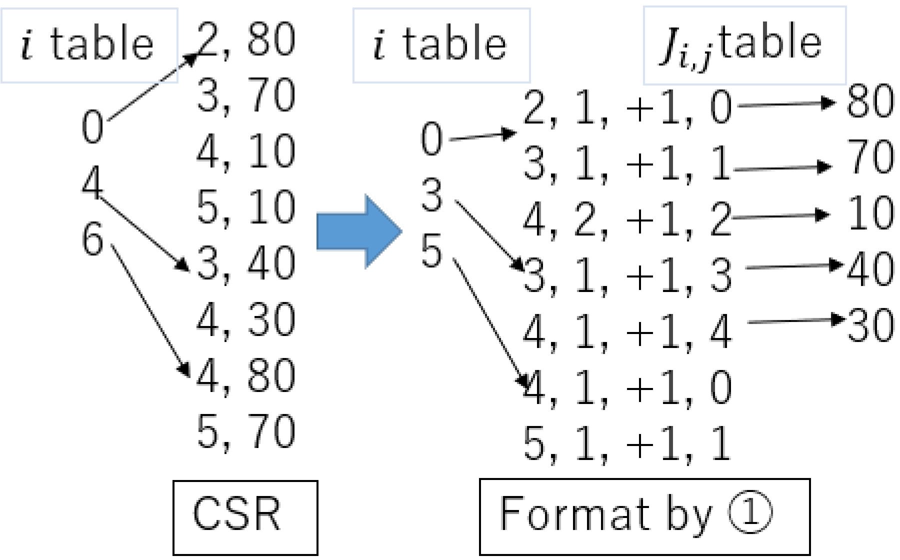


Compress from CSR to Proposed format.

① Repetition of $J_{i,j}$

Focus on repetition $J_{i,j}$ and use list

- Create a table of J_{ij}
- If J_{ij} takes the same value under j changes regularly and the change in j is regular, the number of iterations and the index to the J_{ij} list is used



Compress from COO to ①.

(4, 2, +1, index to 10, 0)
→ 4 is top of j, repeat 2 times, add +1 to j in the repetition j, index to 10

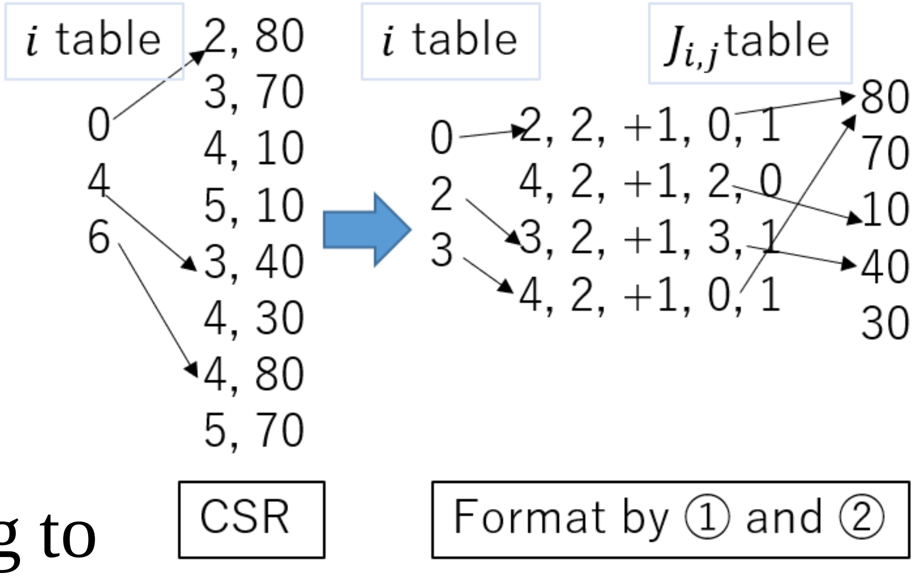
② Value Sequence of $J_{i,j}$

Focus on value sequence of $J_{i,j}$ and use list

- If j changes regularly and $J_{i,j}$ takes a sequence of values, create a continuous J_{ij} list
- Introduce the increment flag and it means tha advance the index of J_{ij} list or not

(2, 2, +1, index to 80, 1)
→ 2 is top of j, repeat 2 times, add +1 in the repetition of j, index to 80, the last 1 is the flag to increment the index of $J_{i,j}$ list

- When the same sequence appears, we can use the same sequence as 4-th line (index to 0)



Compress from COO to ① and ②.

③ Continuity of *j*

Reduce the information of *j*

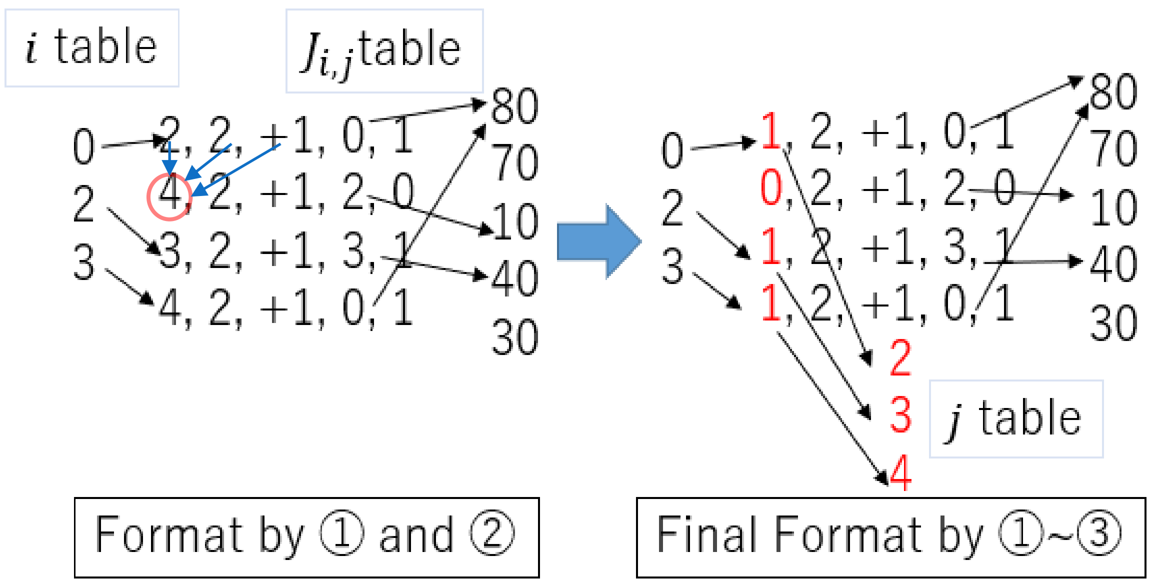
- When the values of *j* are continuous in the same *i*, they may be obtained from the data in the previous row of data

Example: in each lines,

- 2, 2, +1 means 2, 3 and the next value is expected as 4 and matched to the exact value 4
- 4, 2, +1 is 4, 5 and the next is expected as 6 but not matched to the exact value 3

For the not continuous case,

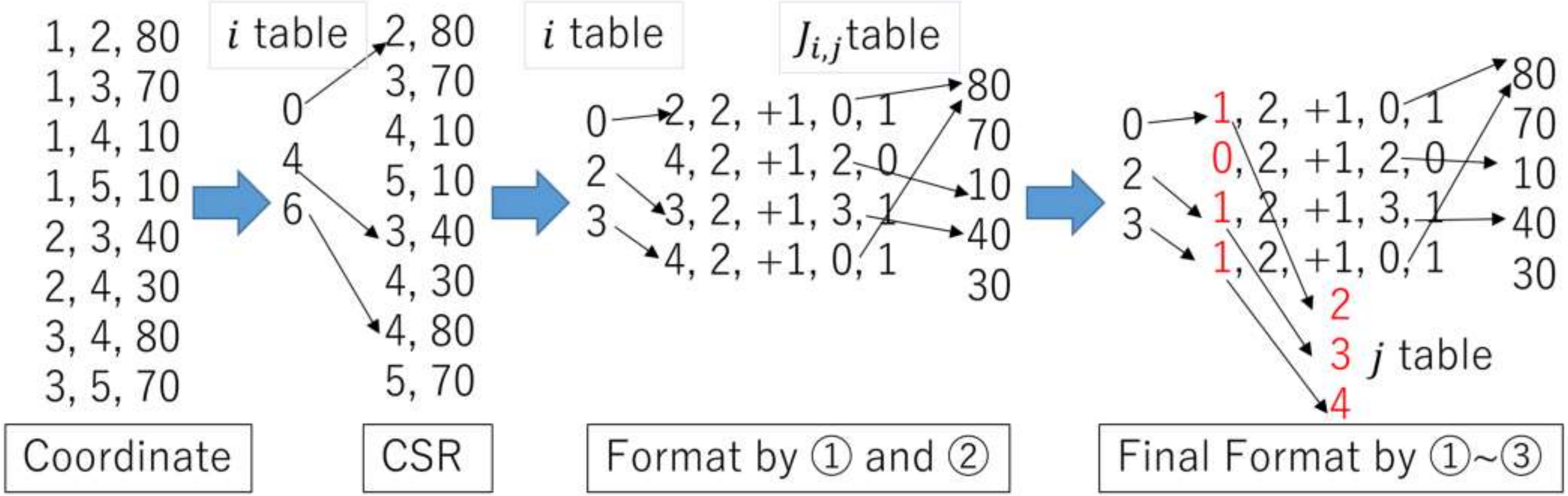
- provide a table to store the first value of *j*
- *j* value is represented by a flag to refer *j* table



Compress from ① and ② to ①~③.

Compare of COO, CSR and Proposed Formats

Bit width of indexes and flag variables can be set to reduce data size



- Note that the bit width of a flag is 1 bit, and an index can be set properly like 16 bit or so depending on data
- The proposed format is effective for large data as shown by experiments

Evaluation Items

Data size and simulation time on SQA is evaluated on SQA

① Data sizes are compared on TSP using file size

② Data sizes are compared on TSP considering bit width

- Compare data size between COO format, CSR and proposed format
- COO and CSR format only includes data $J_{i,j}$ for only $i < j$ and proposed format includes $J_{i,j}$ for $i < j$ and $i > j$ → The compression ratio with respect to the duplicated data is about 1/2 of the following evaluation

③ Effect of Compressed Data on SQA Simulation time

- The SQA simulator/emulator that uses compressed data directly is developed
- SQA simulator/emulator using CSR is also made and compared

① Data Size Compression on TSP

Significant compression effect compared to CSR method confirmed

- Compress 3 input data of 32 city TSP and each 1 input data of 64, 96 city TSP
- The compression time of each problem is less than 1 second on all samples

Compare with COO (Before): 1/12~1/40
Compare with CSR: 1/10~1/30

- The more the number of cities, the higher the compression effect

Compression result of TSP.

	city size	Variables	Before[Bytes]	CSR[Bytes]	After[Bytes]	After/Before
ex32_1	32	1024	1,226,468	972,786	105,371	8.591%
ex32_2	32	1024	1,232,484	980,018	99,435	8.068%
ex32_3	32	1024	1,161,573	923,826	96,036	8.268%
ex64	64	4126	10,864,613	8,429,264	421,540	3.880%
ex96	96	9216	37,337,894	28,686,527	966,786	2.589%

② Data Size Compression on TSP Considering Bit Width

Appropriate bit width is used for flag and index values

- Previous page: comparison of file size (Bytes)
- Here page: the comparison with the number of bits
  - For example, integer corresponds to 32 bits (4 bytes)
  - Flag variables taking 0 or 1 are just 1 bit
- By this, the memory size can be reduced on emulator
- Set the bit width that can correspond to the input data of 96 city TSP

Compare with COO (Before):
1/13~1/40
Compare with CSR:
1/10~1/30

Can apply to hardware emulation

Compression result of TSP considering bit width.

	Variables	Before[Bits]	CSR[Bits]	After[Bits]	After/Before
ex32_1	1024	4,096,000	3,063,808	309,056	7.545%
ex32_2	1024	4,096,000	3,063,808	309,280	7.551%
ex32_3	1024	4,096,000	3,063,808	307,968	7.519%
ex64	4096	33,161,216	24,838,144	1,242,208	3.746%
ex96	9216	112,361,472	84,197,376	2,797,280	2.490%

③ Effect of Compressed Data on SQA Simulation Time

1.9 times faster on CPU and 1.4 times faster on FPGA w.r.t. CSR

Parameter of SQA simulator.

Item	Numeric
β	10.0
Γ	1.0
Trotter	32
Outer loop	1000
Inner loop	100000

Simulation time on CPU.

	Variables	CSR-base[s]	Propose[s]
ex32_1	1024	38.174	27.988
ex32_2	1024	38.086	27.809
ex32_3	1024	38.067	27.266
ex64	4096	81.176	48.420
ex96	9216	143.882	75.751

◆Developed the SQA simulator that uses compressed data directly on CPU

- Intel Core i9-7900XCPU@3.30GHz, 128 GB Memory

- On 96 city TSP, achieve 1.9 times faster
  - This effect would be from the reduction of memory access

◆Developed the SQA emulator on FPGA (Xilinx Alveo U250)

- On CSR-based SQA, can't be developed ×
  → Cause of memory in FPGA
- Proposed-based SQA can be developed 96 city TSP
- On 32 city TSP, achieve 1.4 times faster

Conclusion

Propose a new data format for sparse matrix input data

- The repetition of the same value and the same sequence on the input data is used
- The independent value table is introduced to perform data compression
- Achieve the size reduction 1/40 than COO and 1/30 than CSR on 96 city TSP
- The proposed method based SQA could solve 96 city TSP 1.9 times faster on CPU and 1.4 times faster on FPGA than CSR
- The proposed format-based SQA emulator can run for 64 and 96 city TSP that become out of memory in the CSR-based emulator
- Application to other combinatorial problems is one of future works