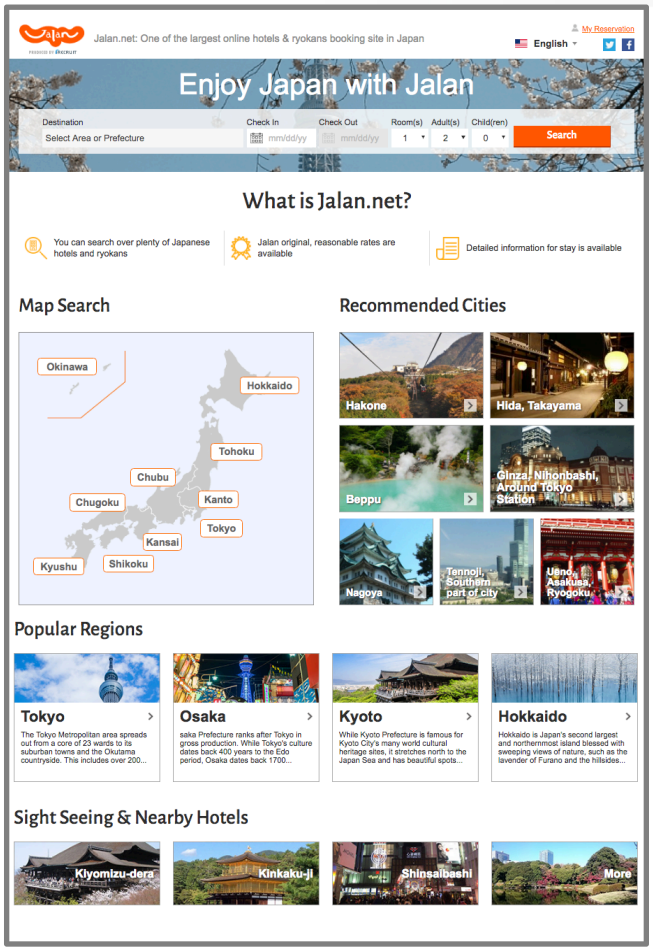
Qubits North America 2018

Item Listing Optimization Considering Diversity in E-Commerce Websites

Naoki Nishimura
Recruit Communications Co., Ltd.
nishimura@r.recruit.co.jp

RECRUIT
Recruit Communications

Introduction of Recruit Group



- Recruit Group provides various kinds of online services - from job search to online shopping - across the globe.
- Examples: Travel reservation, Restaurant reservation, Housing information sites, etc…

Housing	Life & Local O2O	Bridal & Baby
Beauty	Education	Human Resources
Travel	Automobile	IT & Trends Media
		Dining

- Today's topic is the use case of D-Wave on the hotel reservation site "Jalan"
  - https://www.jalan.net/en

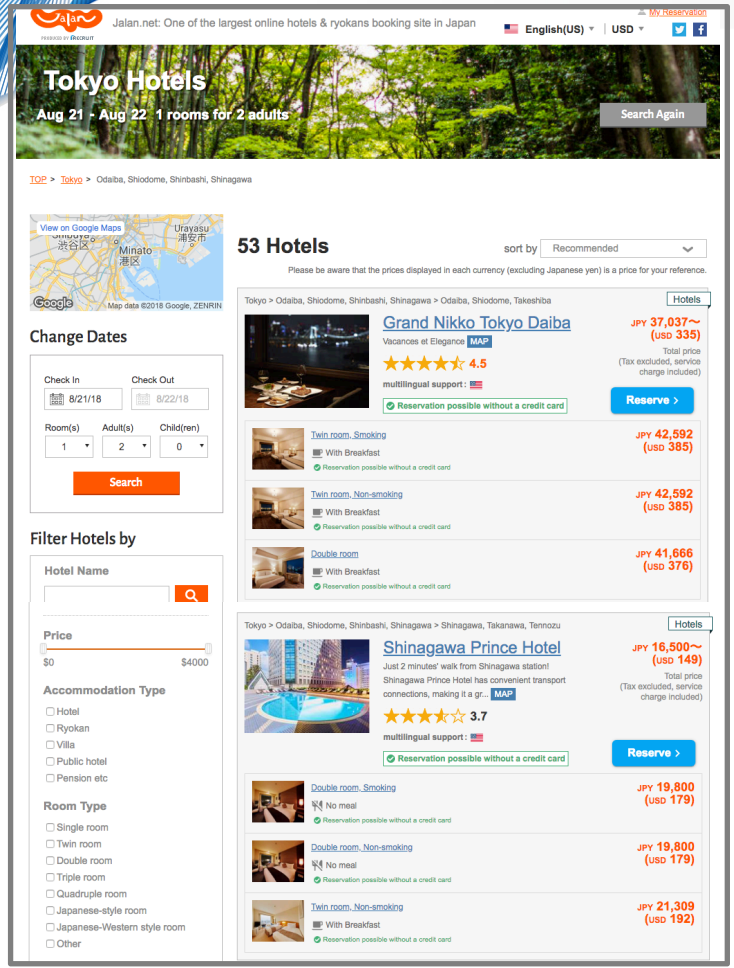
Outline

- Problem Setup: How to list items on an e-commerce website
- Problem Formulation
- Numerical Experiments
- Summary

Outline

- **Problem Setup : How to list items on an e-commerce website**
- Problem Formulation
- Numerical Experiments
- Summary

How to list items on an e-commerce website

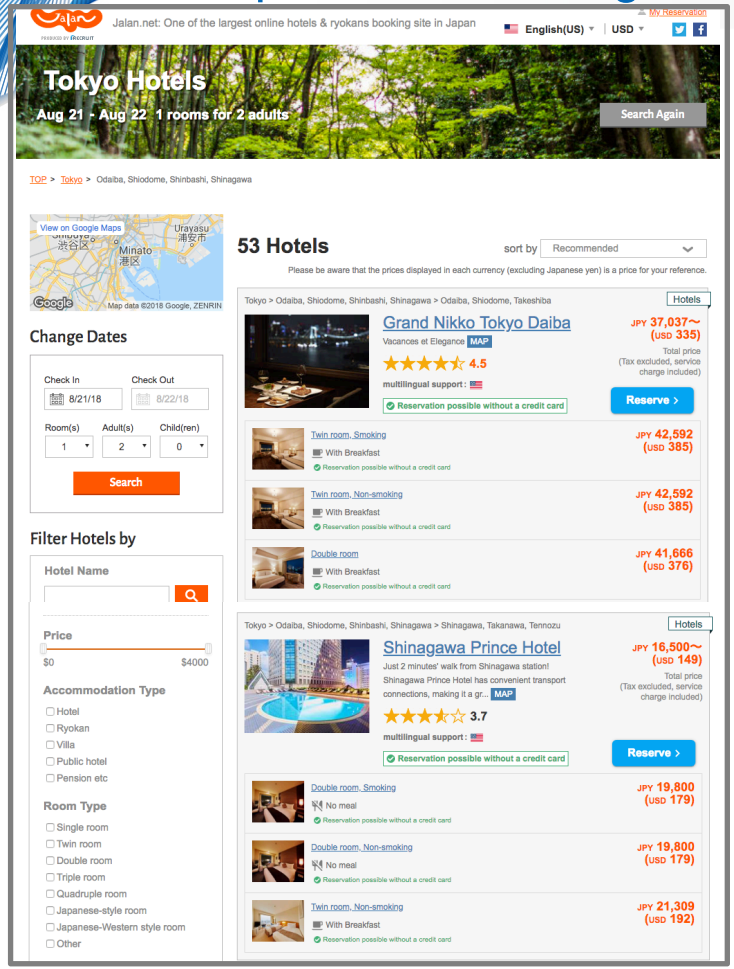


1st list item

2nd list item

- The problem is how to list items on an e-commerce website
- Lists are created for each search segment
  - Area (Tokyo, Kyoto, Osaka …)
  - Number of people to stay (1 per., 2 per., …)
- Lists are created every day based on an algorithm designed to maximize sales

Importance of listing items in e-commerce



High click rate

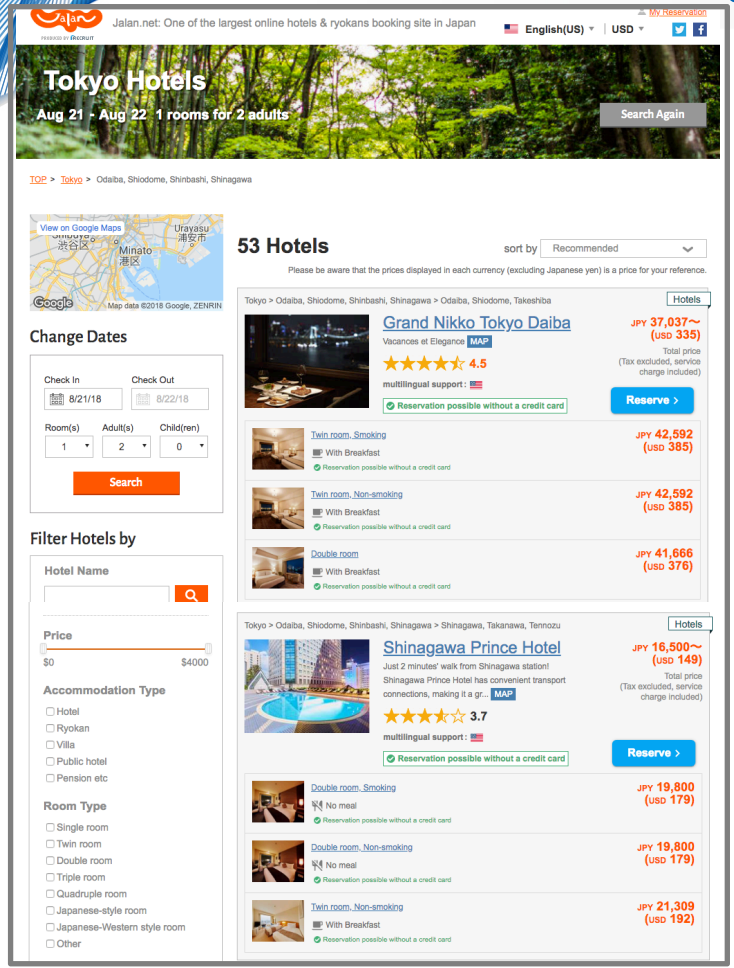
Low click rate

- High percentage of people reserve hotels via the hotel list screen

=> Changing this list has a huge impact

- The higher the item is ranked in the list, the greater the click rate
- Performance difference of about 5% observed between a well-ordered list and randomly ordered list

Considerations in making the item list



In order to create an item list that maximizes sales …

1. Rank items with high sales potential in higher positions on the list

=> Make it easy to find popular items

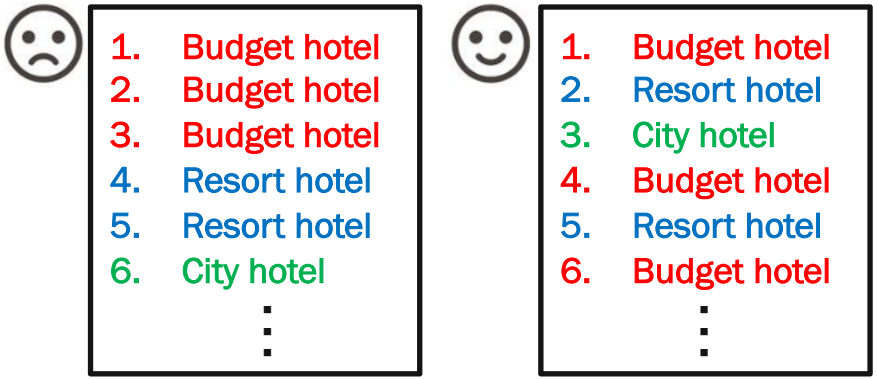
2. Emphasize a certain amount of **diversity** in the items in high ranked positions

=> Make customers aware that they have a wide range of options

What is ‘diversity’ in the item list

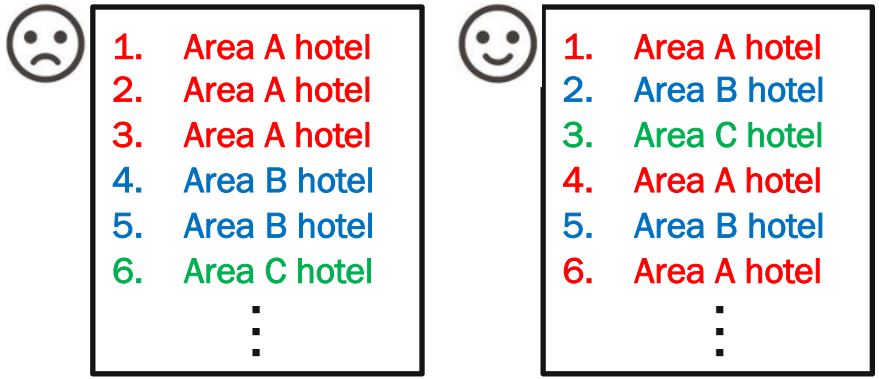
- An example of diversity in the item list

Hotel classification diversity



= Better to have items from various classifications in higher positions

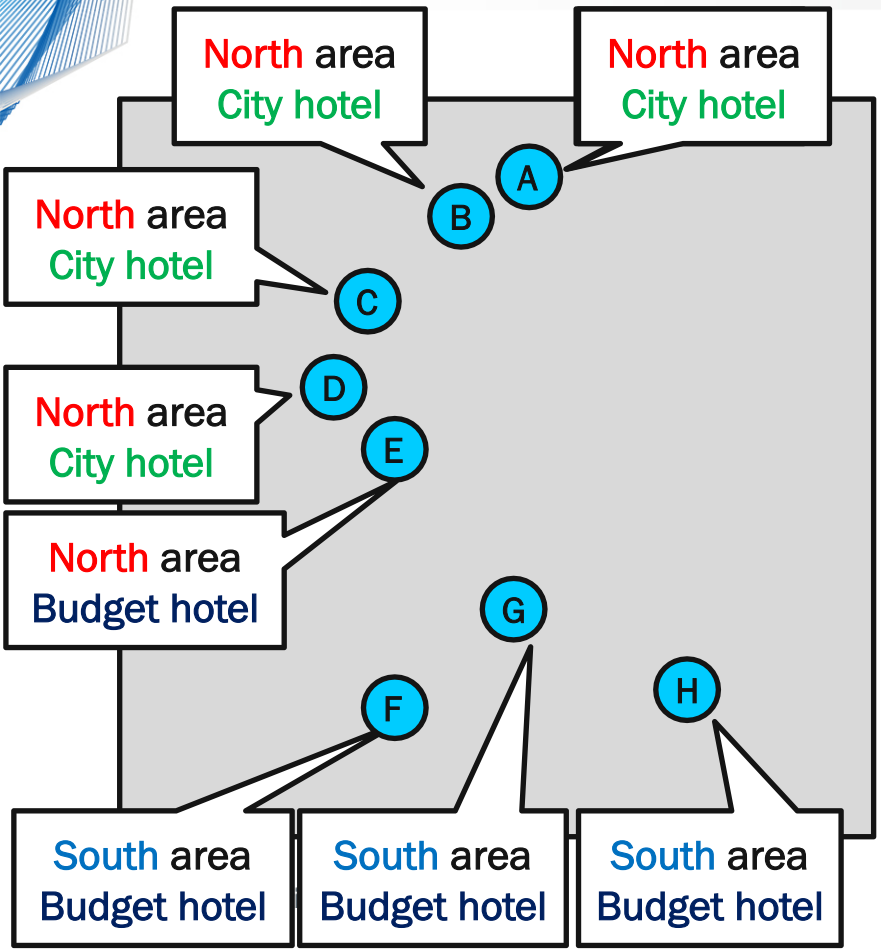
Hotel location diversity



= Better to have items from various locations in higher positions

- Related work on diversity in recommendations:
  - **“Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems” [Vargas et al. ’11]**
  - **“Post Processing Recommender Systems for Diversity” [Antikacioglu et al. ’17]**

Result of solving the problem considering diversity with D-Wave



Solving the problem considering diversity with D-Wave, we got an item list reflecting **both scores and diversity**

=> +1% sales uplift

Item list considering only score

1. C, North, City hotel
2. B, North, City hotel
3. A, North, City hotel
4. D, North, City hotel
5. E, North, Budget hotel
6. H, South, Budget hotel
7. F, South, Budget hotel
8. G, South, Budget hotel

Item list considering diversity

1. B, North, City hotel
2. G, South, Budget hotel
3. A, North, City hotel
4. E, North, Budget hotel
5. H, South, Budget hotel
6. C, North, City hotel
7. F, South, Budget hotel
8. D, North, City hotel

Outline

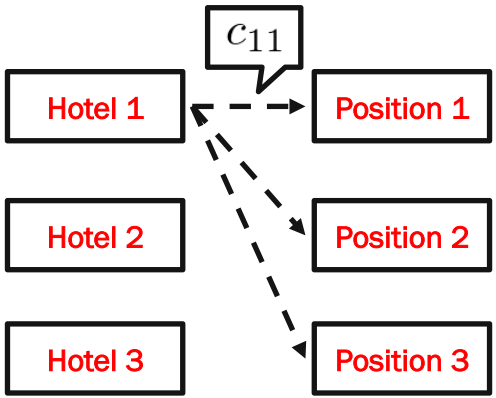
- Problem Setup : How to list items on an e-commerce website
- **Problem Formulation**
- Numerical Experiments
- Summary

Determining the item list considering items' sales scores

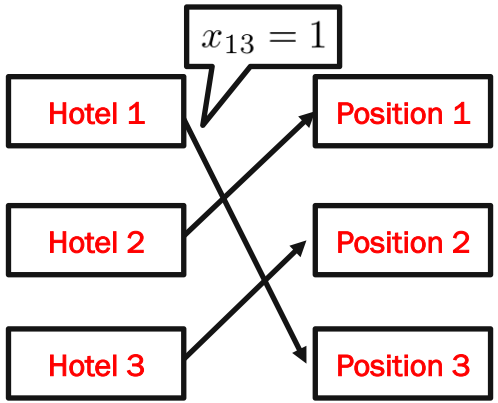
- We formulated the problem of determining the item list considering item sales scores as an Assignment Problem (AP)
- This problem can be solved easily with a general-purpose optimization solver

How to allocate items to each position?

1. Estimate sales when items allocated to each position



2. Decide items' allocation to maximize sales by solving AP



Formulation of AP

$$\max. \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$ [Maximize total sales]

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1, \ \forall j \in J,$$ [One position allocated to only one item]

$$\sum_{j \in J} x_{ij} = 1, \ \forall i \in I,$$ [One item allocated to only one position]

$$x_{ij} \in \{0, 1\}.$$ [Whether item *i* allocated position *j*]

c_{ij} : Sales when item *i* allocated position *j*

x_{ij} : Binary variable whether item *i* allocated position *j*

I : Set of items

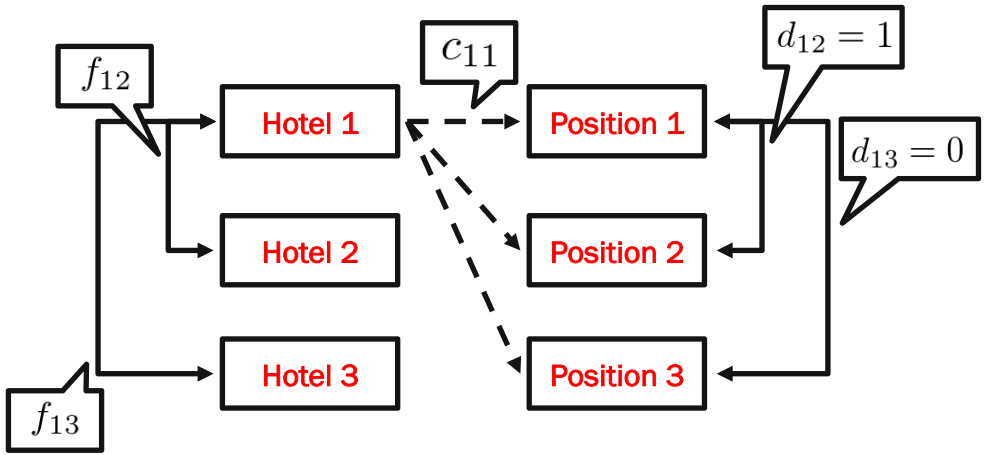
J : Set of positions

Determining item list considering items' diversity

- We formulated the problem of determining the product list considering item sales' scores and diversity as a Quadratic Assignment Problem (QAP) → **NP-hard problem**

How to allocate items to each position?

1. Estimate sales when items allocated to each position and item similarity (customer pageview score)



2. Decide allocation of items to maximize score by solving QAP

Formulation of QAP

$$\max. \underbrace{\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}}_{\text{[item sales' score]}} - w \underbrace{\sum_{i\in I}\sum_{i'\in I}\sum_{j\in J}\sum_{j'\in J} f_{ii'}d_{jj'}x_{ij}x_{i'j'}}_{\text{[penalty of less item diversity]}}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \ \forall j \in J,$$

$$\sum_{j\in J} x_{ij} = 1, \ \forall i \in I,$$

$$x_{ij} \in \{0, 1\}.$$

$f_{ii'}$:	Similarity between item *i and i'* (Number of pageviews in the same session)
$d_{jj'}$:	Closeness between position *j* and *j'* (When they are next to each other 1, else 0)
w :	Control parameter of diversity term

Convert QAP problem into QUBO problem

- We transformed the QAP problem into Quadratic Unconstrained Binary Optimization (QUBO) problems

Formulation of QAP

$$\max. \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} - w\sum_{i\in I}\sum_{i'\in I}\sum_{j\in J}\sum_{j'\in J} f_{ii'}d_{jj'}x_{ij}x_{i'j'}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \ \forall j \in J,$$

$$\sum_{j\in J} x_{ij} = 1, \ \forall i \in I,$$

$$x_{ij} \in \{0,1\}.$$

Formulation of QUBO

$$\min. \quad -\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + w\sum_{i\in I}\sum_{i'\in I}\sum_{j\in J}\sum_{j'\in J} f_{ii'}d_{jj'}x_{ij}x_{i'j'} + \sum_{j\in J} M\left(\sum_{i\in I} x_{ij} - 1\right)^2 + \sum_{i\in I} M\left(\sum_{j\in J} x_{ij} - 1\right)^2$$

[Penalty of constraint violation]

[Penalty of constraint violation]

$$\text{s.t.} \quad x_{ij} \in \{0,1\}.$$

M : Control parameter of constraint violation degree

Outline

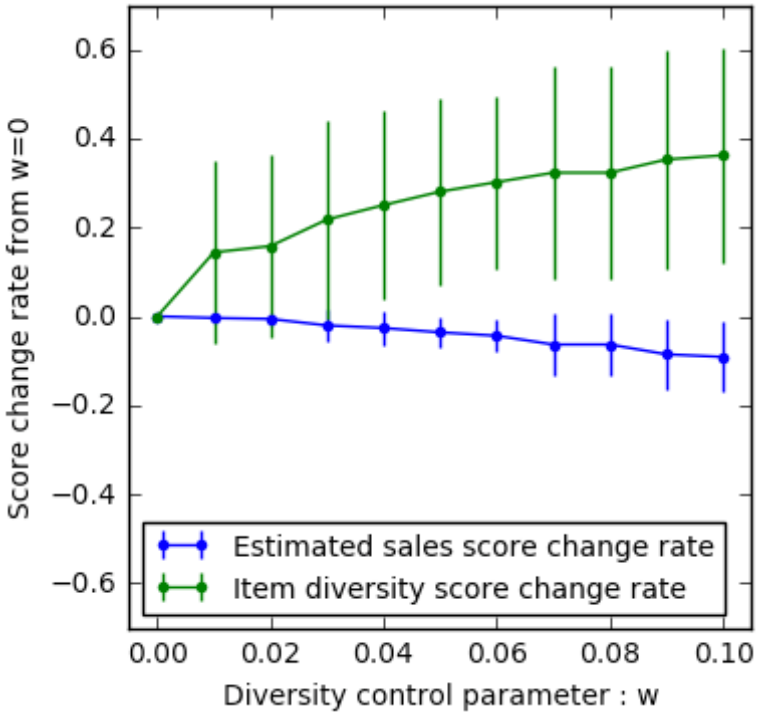
- Problem Setup : How to list items on an e-commerce website
- Problem Formulation
- **Numerical Experiments**
- Summary

Experimental setting

- **Datasets**
  - Top 10 sales areas' pageviews and reservation logs on hotel reservation site "Jalan"
- **Contents**
  1. Change of solution when diversity control parameter is moved
  2. Computing time to find the optimal solution
  3. Distribution of objective values obtained by D-Wave
  4. Comparison of objective values for large problems
- **Computing environment and parameter setting**
  - D-Wave : solver = DW_2000Q_VFYC_2, num_reads=10000, postprocess=optimization, num_spin_reversal_transforms = 4, annealing_time = 20
  - CPLEX : Version 12.6.3, mip.tolerances.mipgap=0, threads=1, CPU 3.1 GHz Intel Core i7, RAM 16GB

Comparison of solutions when changing diversity control parameter

Score change rate by controlling w



- When the parameter *w* increased
  - Estimated sales score decreased
  - Item diversity score increased

$$\max. \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} - w\sum_{i\in I}\sum_{i'\in I}\sum_{j\in J}\sum_{j'\in J} f_{ii'}d_{jj'}x_{ij}x_{i'j'}$$

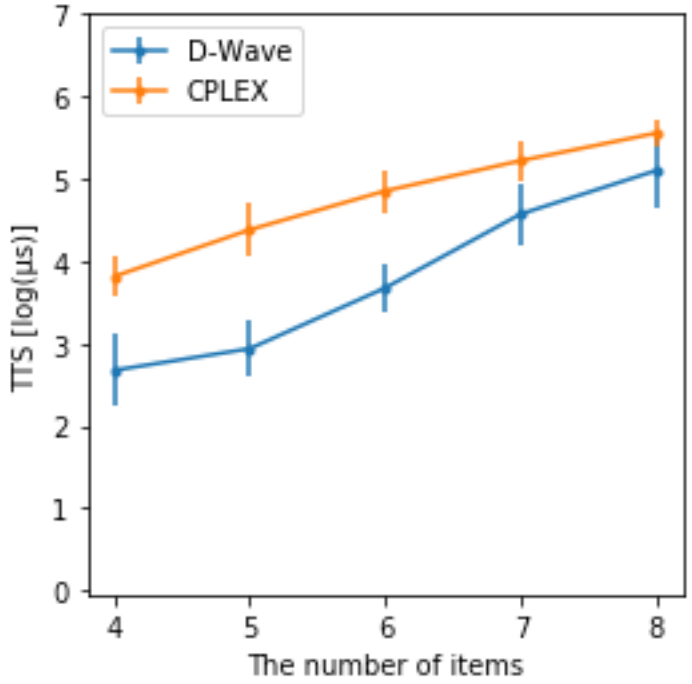
[Estimated sales score] [Item diversity score]

- After this slide, the parameter *w* will be set to 0.07 for considering diversity

RECRUIT
Recruit Communications

Computing time to find the optimal solution

Comparison of TTS (Time-To-Solution)



- In terms of TTS (Time-To-Solution), D-Wave finds the optimal solution faster than CPLEX in our problems

- The definition of TTS [Rønnow, T. F. et al. '14]

$$\mathrm{TTS}(t_f) = t_f R(t_f)\,, \quad R(t_f) = \frac{\ln(1-p_\mathrm{d})}{\ln[1-p_\mathrm{S}(t_f)]}$$

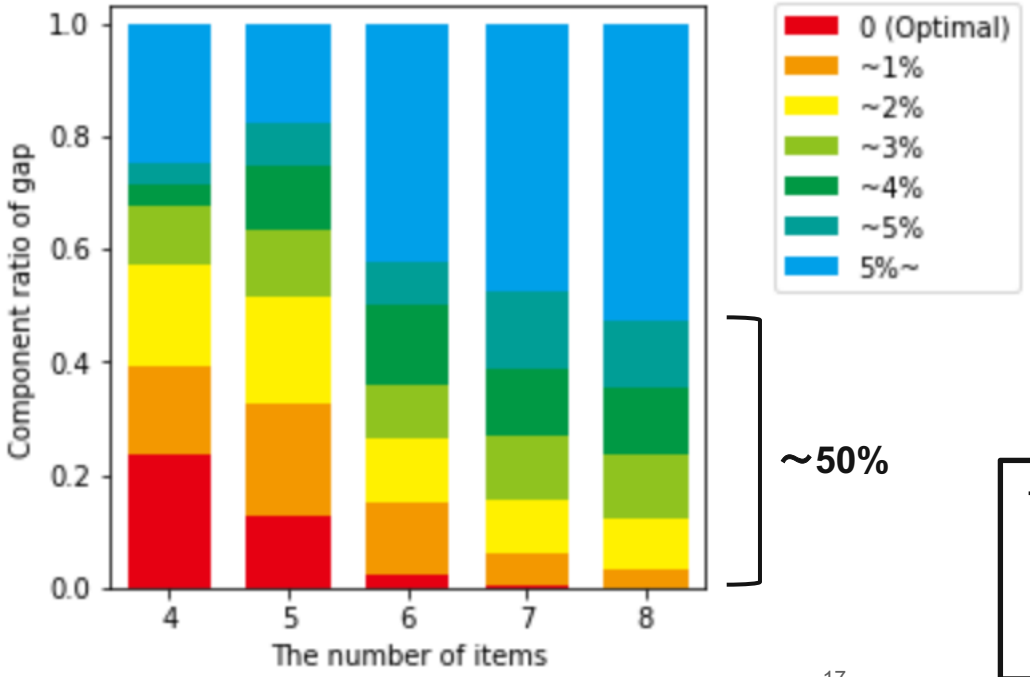
t_f :	Runtime
$R(t_f)$:	Required number of runs to find optimal solution
$p_S(t_f)$:	Success probability of a single-instance run of the algorithm with a runtime
p_d :	Some desired probability (set to 0.99)

RECRUIT
Recruit Communications

* D-Wave solved QUBO problems, and CPLEX solved QAP problems that have constraints

Distribution of objective values obtained by D-Wave

Distribution of the gap from optimal value



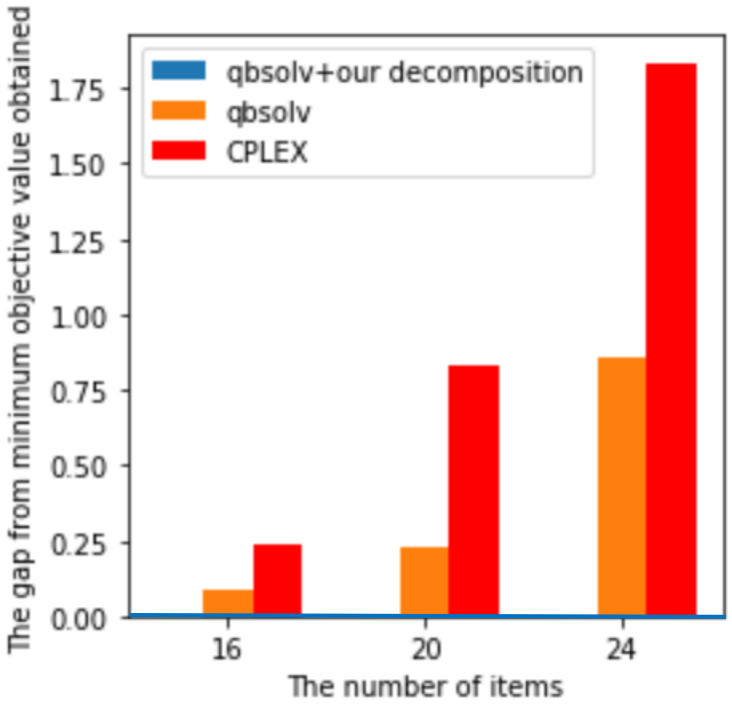
- As the number of items increases, the rate of obtaining optimal solutions decreases
- When the number of items is 8, about 50% of solutions have a gap from optimal value within 5%

The definition of Gap:

$$\mathrm{GAP}=\frac{Optimal\ value - Obtained\ value}{Optimal\ value}$$

Comparison of objective values for large problems in same computing time

Gap from minimum obtained value



- We proposed a decomposition method in qbsolv for our problem
- Our proposed model could find better solutions than other models
- The gap between our qbsolv object value and the other object value increases as the size of the problem became large

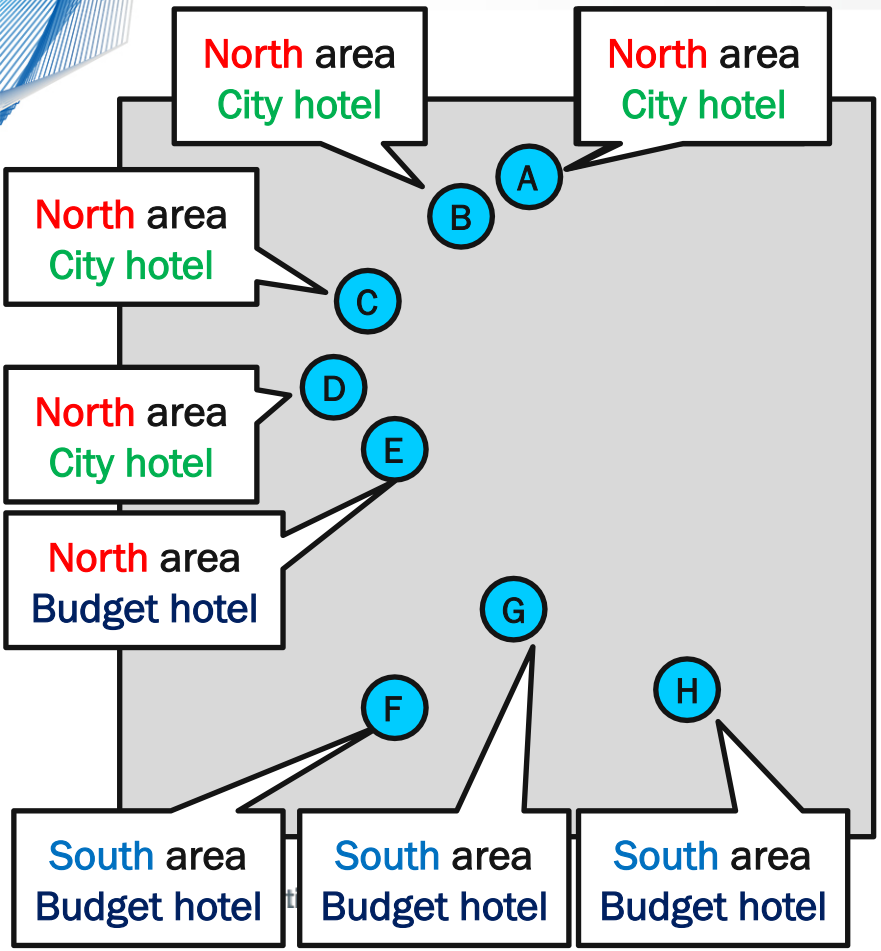
The definition of Gap:

$$\mathrm{GAP}=\frac{Minimum\ obtained\ value - Obtained\ value}{Minimum\ obtained\ value}$$

* Time limit of CPLEX is set to the time required by qbsolv

RECRUIT
Recruit Communications

Interpretation of solution obtained by solving QAP



By solving the problem, we got item list which reflects **both scores and diversity**

Item list considering only score
(w=0, item diversity score = 1)

1. C, North, City hotel
2. B, North, City hotel
3. A, North, City hotel
4. D, North, City hotel
5. E, North, Budget hotel
6. H, South, Budget hotel
7. F, South, Budget hotel
8. G, South, Budget hotel

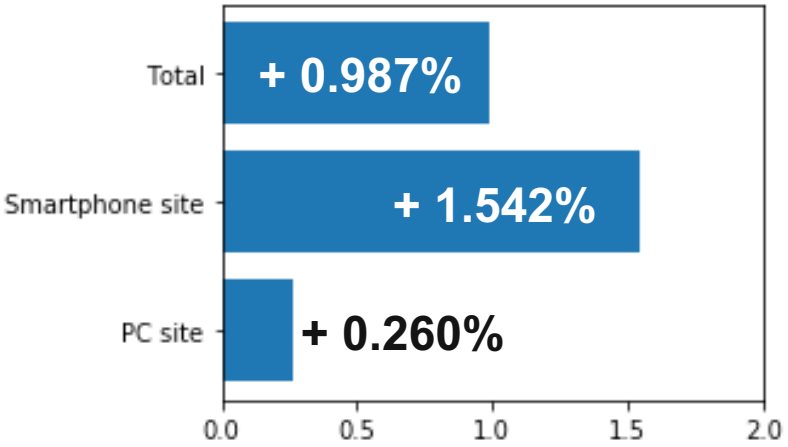
Item list considering diversity
(w=0.07, item diversity score = 1.27)

1. B, North, City hotel
2. G, South, Budget hotel
3. A, North, City hotel
4. E, North, Budget hotel
5. H, South, Budget hotel
6. C, North, City hotel
7. F, South, Budget hotel
8. D, North, City hotel

The result of practical AB testing on our site

Sales uplift considering both sales and diversity

Result of AB testing from Aug1 to Sep10



- In AB testing, we observed better performance considering **both sales and diversity** than considering sales alone
  - **Total sales uplift -> + 0.987%**
- Considering diversity is especially important in smartphone sites
- Future work
  - Adjustment of the diversity parameter in real AB testing

RECRUIT
Recruit Communications

Outline

- Problem Setup : How to list items on an e-commerce website
- Problem Formulation
- Numerical Experiments
- **Summary**

Summary

- We formulated the problem of determining the item list considering item **scores** and **diversity** as a Quadratic Assignment Problem (QAP), which is an NP-hard problem
- We transformed the problem into Quadratic Unconstrained Binary Optimization (QUBO) problems
- We confirmed that the order of the list determined reflects both scores and diversity

Collaborator

This work was done in collaboration with:

- Masayuki Ohzeki (Tohoku University)
- Masamichi J. Miyama (Tohoku University)
- Kotaro Tanahashi (Recruit Communications Co., Ltd.)
- Koji Suganuma (Recruit Communications Co., Ltd.)

Appendix: A Smart Decomposition Method for Assignment Problem

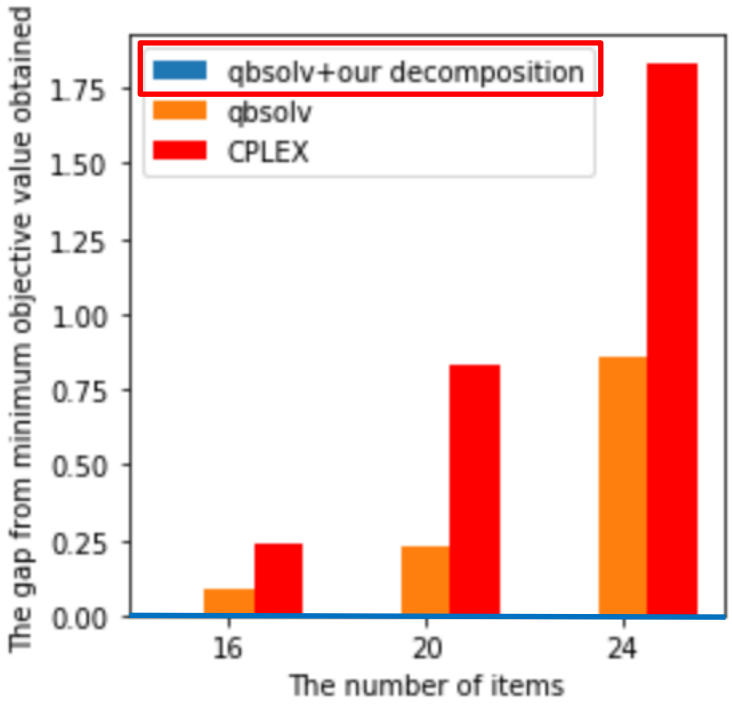
Kotaro Tanahashi, Naoki Nishimura

Recruit Communications Co.,Ltd.

RECRUIT
Recruit Communications

Comparison of Objective Values for Large Problems with the Same Computing Time

Gap from minimum obtained value



* Time limit of CPLEX is set to the time required by qbsolv

We only modified the decomposition method in qbsolv

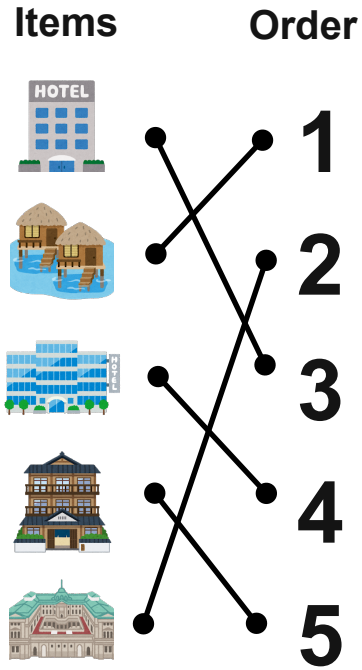
- We proposed a decomposition method in qbsolv for our problem
- Our proposed model could find better solutions than other models
- The gap between our qbsolv object value and the other object value increases as the size of the problem became large

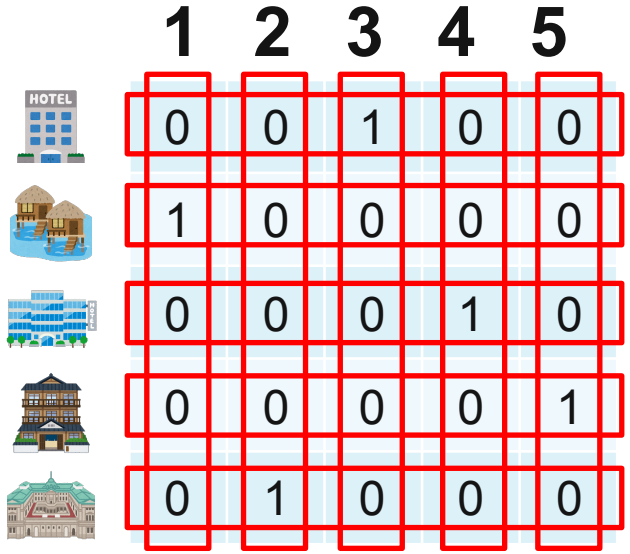
The definition of Gap:

$$\mathrm{GAP}=\frac{Minimum\ obtained\ value - Obtained\ value}{Minimum\ obtained\ value}$$

RECRUIT
Recruit Communications

Binary Representation of Assignment Problem (AP)





Sum of each row is 1
Sum of each column is 1

AP has complex constraint structure

Smart Selection of Subproblem

Original qbsolv

	1	2	3	4	5
HOTEL	0	0	1	0	0
	1	0	0	0	0
	0	0	0	1	0
	0	0	0	0	1
	0	1	0	0	0

Selection of subproblem
without considering the structure

→ No feasible candidate 😵

Our decomposition

	1	2	3	4	5
HOTEL	0	0	1	0	0
	1	0	0	0	0
	0	0	0	1	0
	0	0	0	0	1
	0	1	0	0	0

Selection of subproblem
based on the logical structure

→ **3!-1 feasible candidates** 😊

⇔

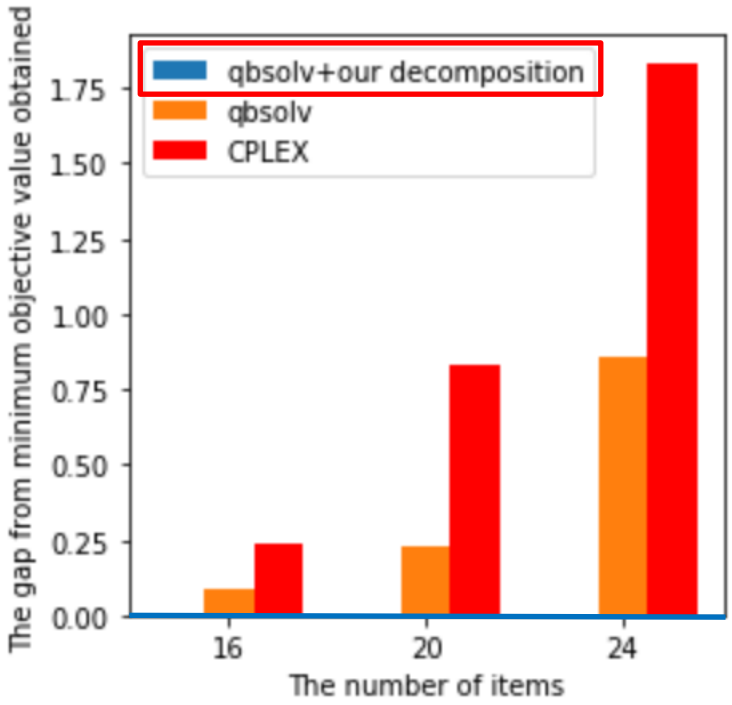
1	HOTEL	Optimize
2		
3		
4		Freeze
5		

Logical meaning

Ex) Size of subproblem = $3^2=9$

Comparison of Objective Values for Large Problems with the Same Computing Time

Gap from minimum obtained value



* Time limit of CPLEX is set to the time required by qbsolv

We modified the decomposition method in qbsolv

- We proposed a decomposition method in qbsolv for our problem
- Our proposed model could find better solutions than other models
- The gap between our qbsolv object value and the other object value increases as the size of the problem became large

The definition of Gap:

$$\mathrm{GAP}=\frac{Minimum\ obtained\ value - Obtained\ value}{Minimum\ obtained\ value}$$

Sep. 25, 2018

Introduction of DSL: PyQUBO for Programing QUBOs in Quantum Annealing

Kotaro Tanahashi*, Shu Tanaka†

* Recruit Communications Co.,Ltd.

† Waseda University, JST PRESTO

RECRUIT

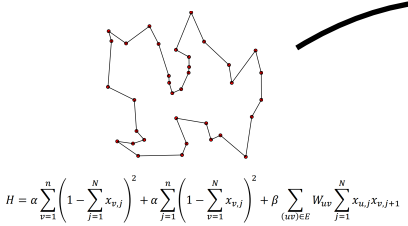
Recruit Communications

How to Embed Problems in Annealing Machines

We need to write a program to construct QUBO for every specific problem.

QUBO formulation ⇔ determine Q_{ij}

Embed in sparse hardware



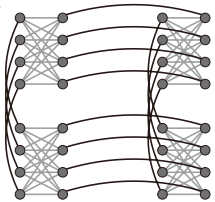
$$H = \alpha \sum_{v=1}^{n} \left(1 - \sum_{j=1}^{N} x_{v,j}\right)^2 + \alpha \sum_{j=1}^{n} \left(1 - \sum_{v=1}^{N} x_{v,j}\right)^2 + \beta \sum_{(uv)\in E} W_{uv} \sum_{j=1}^{N} x_{u,j} x_{v,j+1}$$

Original problem

$$\boldsymbol{x}^* = \min \sum_{i \geq j} x_i Q_{ij} x_j$$

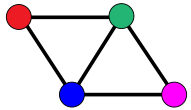
$$\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{Q} \in R^{n \times n}$$

QUBO

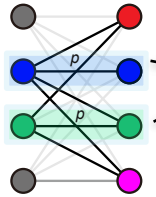


Hardware

Minor Embedding
[J. Cai+ 2014]



dense graph



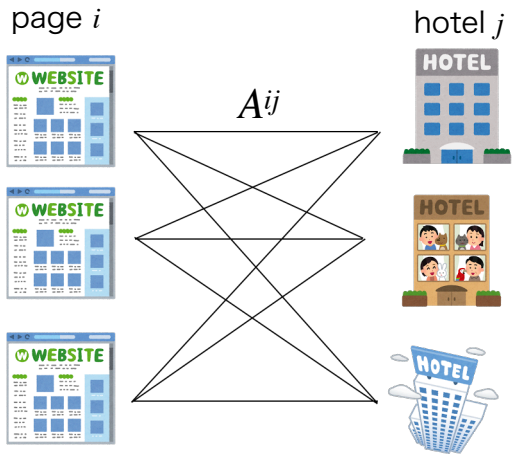
Chain

p: Chain penalty strength

sparse graph

RECRUIT
Recruit Communications

QUBO Construction in Recommendation



- Maximize sum of scores ΣA^{ij}
- Show a hotels in each page
- Each hotel is shown b times

Maximize H - C

$$H = \sum_{i}^{n}\sum_{j}^{m} A^{ij}x_{ij}/2$$

$$C = M\sum_{i}^{n}\left(\sum_{j}^{m} x_{i,j} - a\right)^2 + M\sum_{j}^{m}\left(\sum_{i}^{n} x_{i,j} - b\right)^2$$

$$x_i \in \{0,1\}$$

+ Introduce interaction between edges (QAP)

$$w\sum_{i\in I}\sum_{i'\in I}\sum_{j\in J}\sum_{j'\in J} f_{ii'}d_{jj'}x_{ij}x_{i'j'}$$

+ Increase categorical diversity

$$\sum_{i}\left(\sum_{j\in C_1} x_{ij}\sum_{k\in C_2} x_{ij}\right)$$

Python code to build QUBO

```
coeff = 2.0

def toIndex(i, j):
    return i * m + j

Q = collections.defaultdict(int)
max_Aij = 0
for i in range(n):
    for j in range(m):
        Q[toIndex (i, j), toIndex (i, j)] = -A[i,j]/2
        max_Aij = A[i,j] if max_Aij < A[i,j] else max_Aij

M = max_A * coeff

# quadratic parts in the second terms
for i in range(n):
    for k in range(m):
        for l in range(m):
            Q[toIndex (i, k), toIndex (i, l)] += M

# quadratic parts in the third term
for j in range(m):
    for k in range(n):
        for l in range(n):
            Q[toIndex (k, j), toIndex (l, j)] += M

# linear parts
for i in range(n):
    for j in range(m):
        Q[toIndex (i, j), toIndex (i, j)] += M*(-2*a -2*b)

# constant
c = M*(n*a*a + m*b*b)
```

- Specific for the problem
- Readability is low

RECRUIT
Recruit Communications

Developed DSL for Building QUBOs: PyQUBO

Example: Number partitioning problem with S = {4, 2, 7, 1}

$$H = (4s_1 + 2s_2 + 7s_3 + 1s_4)^2 \quad s_i \in \{-1, 1\}$$

Create QUBO with Domain Specific Language (DSL).

```
>>> from pyqubo import Spin
>>> s1, s2, s3, s4 = Spin("s1"), Spin("s2"), Spin("s3"), Spin("s4")
>>> H = (4*s1 + 2*s2 + 7*s3 + s4)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo()
>>> pprint(qubo)
{('s1', 's1'): -160.0,
 ('s1', 's2'): 64.0,
 ('s1', 's3'): 224.0,
 ('s1', 's4'): 32.0,
 ('s2', 's2'): -96.0,
 ('s2', 's3'): 112.0,
 ('s2', 's4'): 16.0,
 ('s3', 's3'): -196.0,
 ('s3', 's4'): 56.0,
 ('s4', 's4'): -52.0}
```

1. Define the hamiltonian

2. Compile the hamiltonian

3. Call `to_qubo()`

RECRUIT
Recruit Communications

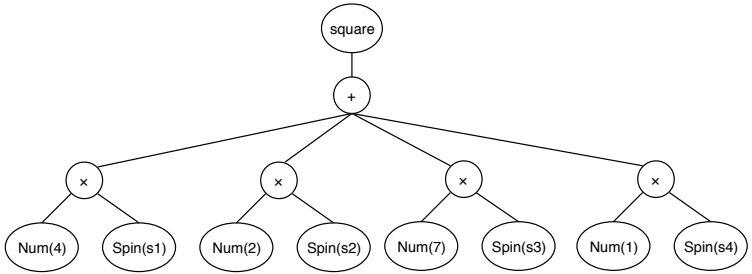
Internal Structure of Expression

Example: number partitioning problem with S = {4, 2, 7, 1}

$$H = (4s_1 + 2s_2 + 7s_3 + 1s_4)^2$$

```
>>> s1, s2, s3, s4 = Spin("s1"), Spin("s2"), Spin("s3"), Spin("s4")
>>> H = (4*s1 + 2*s2 + 7*s3 + s4)**2
```

Internal structure of H



compile

QUBO

```
{('s1', 's1'): -160.0,
 ('s1', 's2'): 64.0,
 ('s1', 's3'): 224.0,
 ('s1', 's4'): 32.0,
 ('s2', 's2'): -96.0,
 ('s2', 's3'): 112.0,
 ('s2', 's4'): 16.0,
 ('s3', 's3'): -196.0,
 ('s3', 's4'): 56.0,
 ('s4', 's4'): -52.0}
```

The expression is internally represented as a tree structure.

Features of PyQUBO

With PyQUBO, you can do …

- Simplify your code with the power of abstraction
- Automatic validation of constraints
- Just In Time (JIT) compile
- Work with dimod[1] seamlessly

[1] https://github.com/dwavesystems/dimod

The Power of Abstraction: Example of Adder

A, B: binary-encoded integer

$$A = \sum_n 2^n A_n$$

$$B = \sum_n 2^n B_n$$

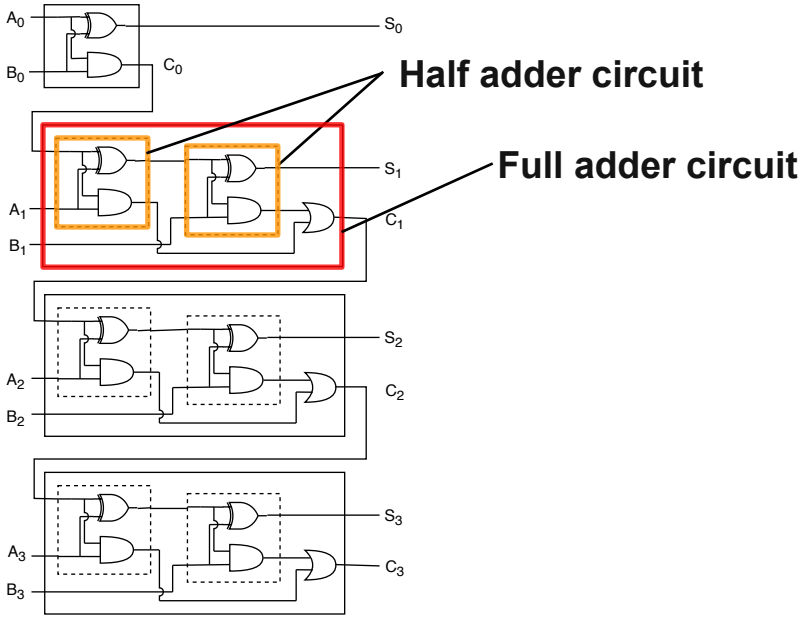
$$S = A + B = \sum_n 2^n S_n$$

Example

$$\begin{aligned} A &= [1,0,1,1] \\ +)\ B &= [0,0,1,0] \\ \hline S &= [1,1,0,1] \end{aligned}$$

How to get QUBO of S_n ?

The circuit seems very complex 😖

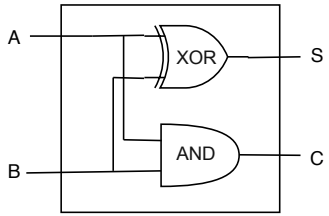


4-bit addition circuit of binary-encoded integer A and B

Recruit Communications

Class of Half Adder and Full Adder

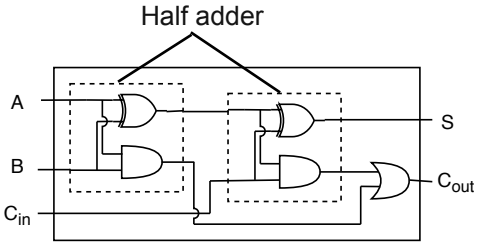
We can define the class of the circuit with DSL.



Half adder

```
class HalfAdder:
    def __init__(self, a, b):
        self.s = Xor(a, b)
        self.c = And(a, b)
```

Define the class of half adder

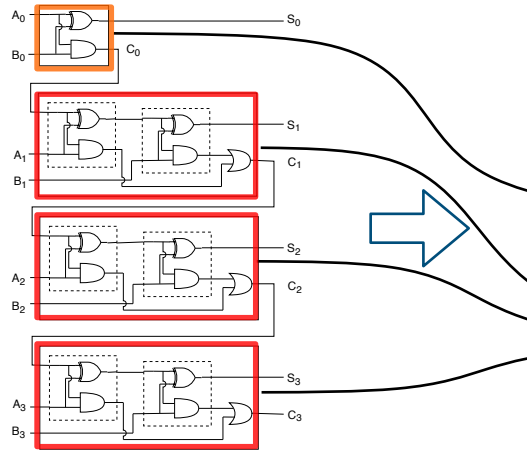


Full adder

```
class FullAdder:
    def __init__(self, a, b, c_in):
        half_adder_1 = HalfAdder(a, b)
        half_adder_2 = HalfAdder(half_adder_1.s, c_in)
        self.s = half_adder_2.s
        self.c_out = Or(half_adder_1.c, half_adder_2.c)
```

Define the class of full adder

Multi-bit Addition



Multi-bit adder

```ruby
class BinaryIntegerVar(UserDefinedExpress):
    ...

    def add(self, other):
        sum_bits = []
        half_adder = HalfAdder(self.bits[0], other.bits[0])
        sum_bits.append(half_adder.s)
        carry = half_adder.c
        for i in range(1, self.n_bits):
            full_adder = FullAdder(self.bits[i], other.bits[i], carry)
            sum_bits.append(full_adder.s)
            carry = full_adder.c_out
        return BinaryIntegerVar(sum_bits)
```

Define the multi-bit addition

Finally, multi-bit addition is simplified as…

```ruby
A = BinaryIntegerVar.new('A', n_bits = 4)
B = BinaryIntegerVar.new('B', n_bits = 4)
S = A.add(B)
S.bits[n].compile().to_qubo()  # <= This is what we want
```

This is just an example.
Create your own class with DSL
And simplify your code.

Automatic Validation of Constraints

You can tell the compiler a constraint part of your hamiltonian

```
Constraint(Hamiltonian of constraint)
```

Example: $H = 2a + b + \boxed{(a+b-1)^2}$ Constraint $a+b=1$

```
>>> a, b = Qbit('a'), Qbit('b')
>>> exp = 2a + b + Constraint((a+b-1)**2, label="one_hot")
>>> model = exp.compile()
```

This part is recognized as a constraint: $a+b=1$

```
# when the constraint is broken
>>> sol, broken, energy = model.decode_solution({'a': 1, 'b': 1}, var_type='binary')
>>> print(broken)
{'one_hot': {'penalty': 1.0, 'result': {'a': 1, 'b': 1}}}
```

When constraint is broken, broken constraint is shown

```
# when no constraint is broken
>>> sol, broken, energy = model.decode_solution({'a': 1, 'b': 0}, var_type='binary')
>>> print(broken)
{}
```

When nothing is broken, `broken` is empty

Just In Time (JIT) Compile

For example, when you solve Traveling Salesman Problem (TSP), you need to tune the penalty strength ***A***.

Hamiltonian of TSP

$$H = \sum_{u}^{n}\sum_{v}^{n} d_{uv} \sum_{j}^{n} x_{j,u}x_{j+1,v} + A\sum_{v}^{n}\left(\sum_{j}^{n} x_{j,v} - 1\right)^2 + A\sum_{j}^{n}\left(\sum_{v}^{n} x_{j,v} - 1\right)^2$$

Distance | Constraint 1 | Constraint 2

- We need to update *A* gradually up to the point where constraints are satisfied.
- If we compile it every time, it takes longer time 😫
- **Can we update *A* without compiling from the beginning?**

Just In Time (JIT) Compile with Param

- Yes, we can. Just define *A* by Param.

Distance Const 1 Const 2

$$H = \sum_u^n \sum_v^n d_{uv} \sum_j^n x_{j,u} x_{j+1,v} + A \sum_v^n \left(\sum_j^n x_{j,v} - 1 \right)^2 + A \sum_j^n \left(\sum_v^n x_{j,v} - 1 \right)^2$$

```
# Define TSP in DSL
x = Matrix('x', n_city, n_city)
distance = Sum(0, n_city, lambda u: Sum(0, n_city, lambda v:
              Sum(0, n_city, lambda j: d(u, v) * x[j, u] * x[(j+1)%n_city, v])))
const_1 = Sum(0, n_city, lambda v: (Sum(0, n_city, lambda j: x[j, v]) - 1)**2)
const_2 = Sum(0, n_city, lambda j: (Sum(0, n_city, lambda v: x[j, v]) - 1)**2)

# Construct hamiltonian and compile it
A = Param('A')
H = distance + A * (const_1 + const_2)
model = H.compile()

# Generate QUBO with different A
for a in [0.1, 0.2, ..., 1.0]
    qubo, offset = model.to_qubo(params={'A': a})
```

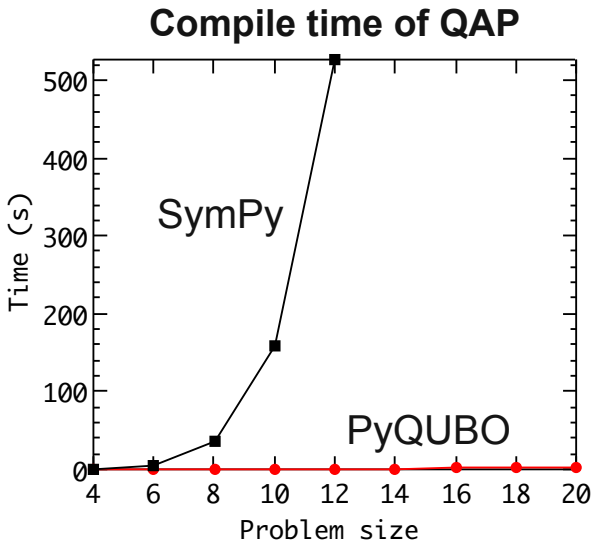
Define *A* by `Param`

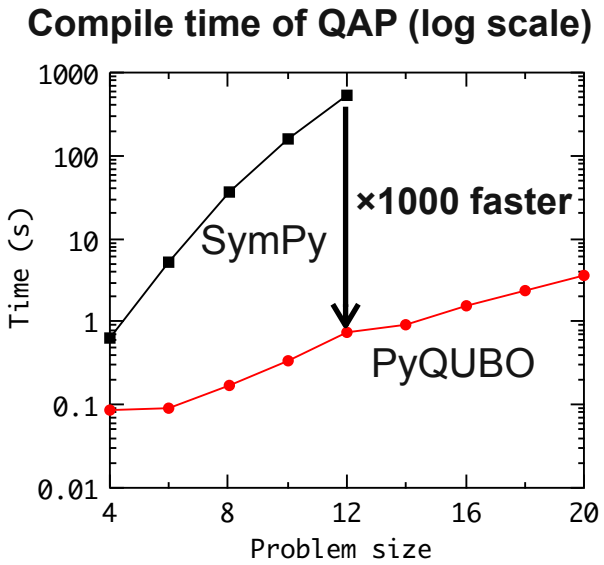
It takes a little time to execute the first compile. `model` **contains half-compiled QUBO.**

You can get QUBO instantly even though you update A

Comparison of Compile Time with SymPy

- You can do the similar operations with SymPy which is a library for symbolic mathematics
- However, PyQUBO is much faster (×1000) than SymPy.

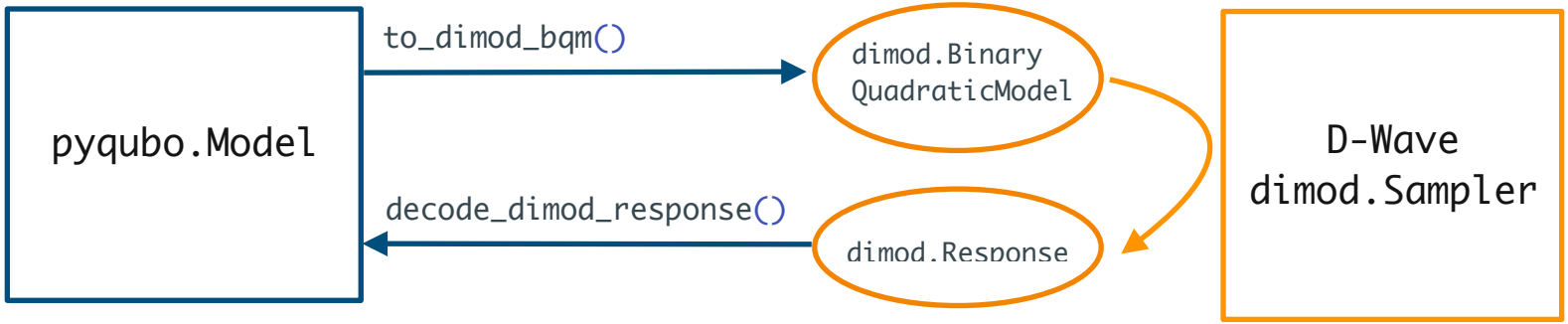




RECRUIT
Recruit Communications

Working with *dimod* Seamlessly

- dimod[1]: a shared API for binary quadratic samplers developed by D-Wave Systems.
- `pyqubo.Model` can export `dimod.BinaryQuadraticModel`.
- `pyqubo.Model` can decode the solution from `dimod.Sampler`

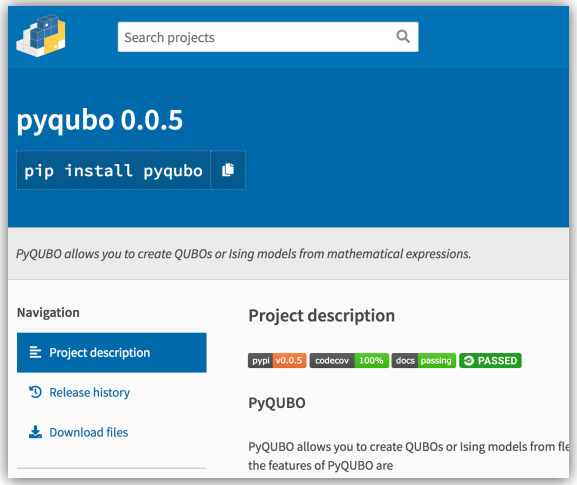


[1] https://github.com/dwavesystems/dimod

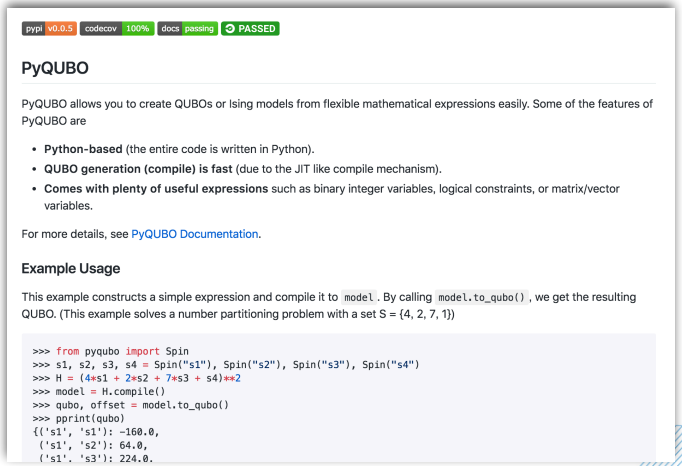
PyQUBO Just Has Been Released 🎉

Please install PyQUBO!!

```
pip install pyqubo
```



https://pypi.org/project/pyqubo/



https://github.com/recruit-communications/pyqubo

Summary

- We developed DSL: PyQUBO for building QUBOs.
- PyQUBO's features are
  - To simplify your code with the power of abstraction
  - Automatic validation of constraints
  - Just In Time (JIT) compile
  - Working with dimod seamlessly
- Enjoy your QUBO life!

Thank you for listening