ACES: Simulated Annealing on Vector Engine (Vector Annealing)

NEC Corporation

Before we begin…

- This presentation is available on the path:

```
/scratch/training/nec/va
```

- The hands-on source codes are available on the path

```
/scratch/training/nec/va/va-demo-codes
```

- Source the script using the following command

```
$ source vasetup
```

- Copy the example codes to your user spaces using the following command

```
$ cp -r /scratch/training/nec/va/va-demo-codes ~
```

…or ask our friends from Texas A&M how to do it using GUI.

Fundamentals of Quantum Mechanics

Theoretical framework describing the behavior of matter and energy at the quantum level.

- It challenges classical intuitions..!
- Introduces concepts like **superposition** and **entanglement** .....among others!
- Introduces probabilistic behavior at the microscopic level, fundamentally altering our understanding of physical phenomena.

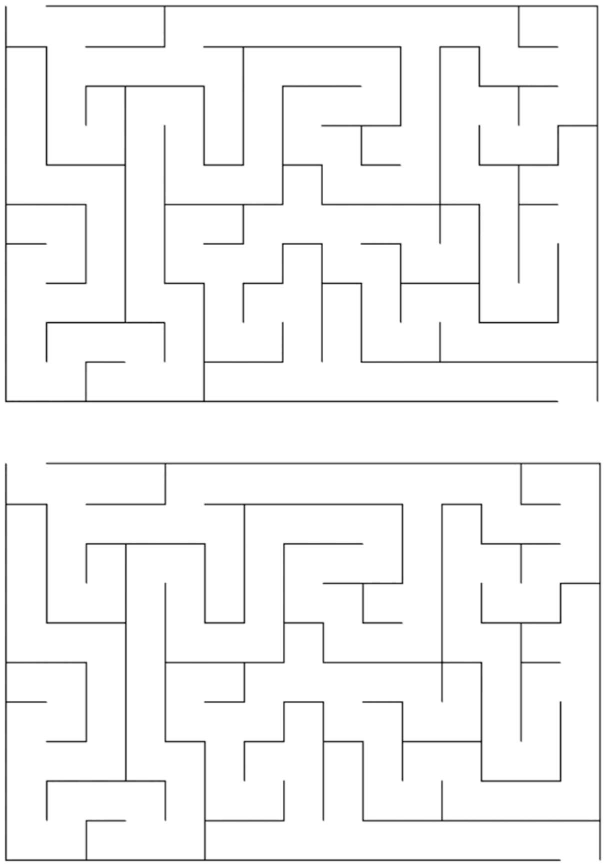
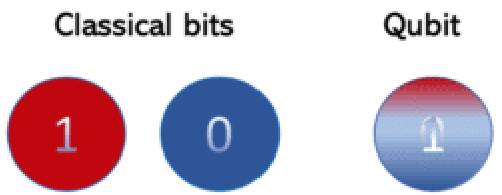


Image credits: https://becominghuman.ai/quantum-superposition-and-what-that-means-to-quantum-computation-3fbb5a711b9a

Bits and Qubits

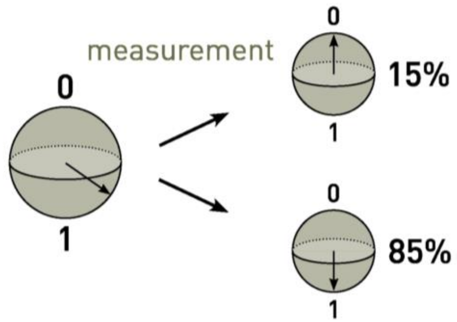
Qubits are the building blocks of quantum computers!

- **Regular bits and Qubits**
  - Regular bits can either 0 or 1
  - Qubits can be 0 and 1 at the same time!

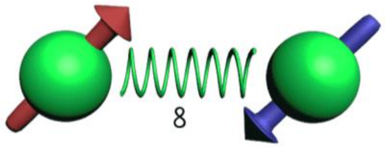


- **Imagine a magical coin**
  - Regular coin can either be Heads or Tails
  - Quantum coin can be in the air – both Heads and Tails, till you catch it.

- **Superposition**
  - Qubits can exist in multiple states simultaneously (like the spinning coin in the air)



- **Entanglement**
  - Qubits can be entangled with each other
  - The state of one qubit is connected to the state of another, irrespective of the distance



Combinatorial Optimization Problems Description

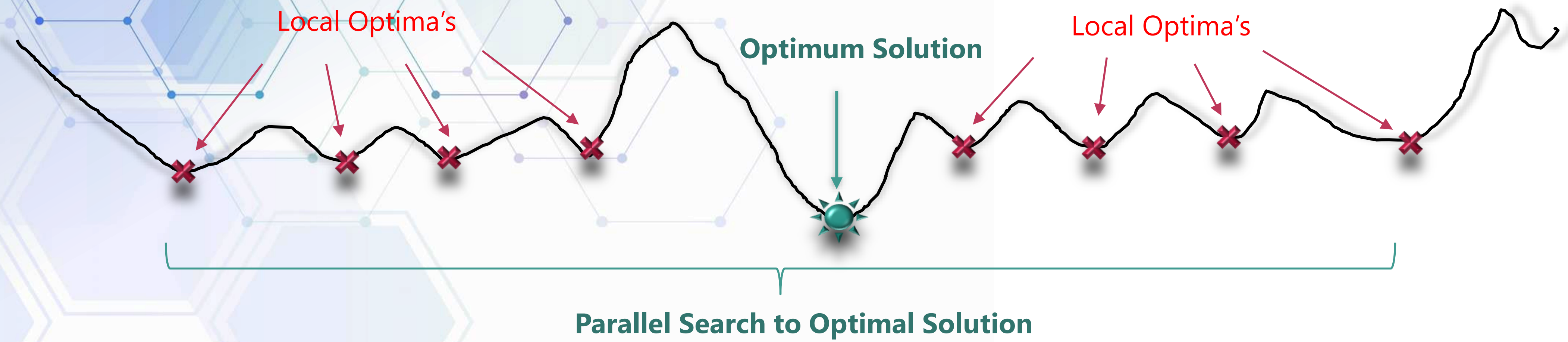
Combinatorial Problem

- Characterized by inputs:
  - an **objective** defining the properties of a solution
  - a general description of **conditions** and **parameters**
- Solved by:
  - Find a **group**, ordering, or assignment of a discrete, finite set of objects that **satisfies** given conditions.
- Combinations of objects or solution components that **need not** satisfy **all given conditions** are **candidate solutions**.
- Candidate solutions that satisfy **all given conditions** are the actual **solutions.**

Optimization Problem

- **Define** an objective function for the inputs.
- Objective function measures solution quality (often defined on all candidate solutions).
- **Minimize/Maximize** the objective function to find a solution with optimal quality.
- Variants of optimization problems:
  - **Search variant**: Find a solution with optimal objective function value for given problem instance.
  - **Evaluation variant**: Determine optimal objective function value for given problem instance.

Combinatorial Optimization Problems



- **Goal:** To find optimal solution or object from a finite set of solutions/space or objects.
- **Challenge:** The solution space is typically too large to search exhaustively using brute force or exploring multiple solution or many local minima's.
- **Examples:** Finding shortest/cheapest round trips(TSP), planning/scheduling, Supply Chain Optimization, Circuit design, Protein Structure Prediction etc.

Quantum Annealing (QA) can explore the solution space in parallel using energy fluctuations - Quantum Effect

Simulated Annealing (SA) simulates the same by a meta-heuristic approach of execution on a classical computer **using Vector Engine accelerators**

Annealing

- "Annealing" is the process of heating a material to critical temperature levels resulting in structural or property changes, followed by cooling it to retain the change.

  Example: *Forging swords*.

- Simulated Annealing (SA) is a probabilistic, meta-heuristic technique inspired from process of annealing metals for solving optimization problems.

- The goal is to achieve minimum energy (entropy), or temperature that results in high probability of success of achieving a speedy as well as accurate solution!

Quantum Annealing

Initialization

- At the start of the quantum annealing process, qubits are initialized in a way that represents the problem at hand.
- These qubits are like the bits in classical computing but with the added magic of quantum superposition and entanglement.

Annealing Schedule

- The system is then subjected to an annealing schedule.
- This schedule defines how the system transitions from an initial state to a final state over time.
- During this evolution, the qubits explore various configurations, seeking the state that corresponds to the solution of the problem.

Objective Function

- Central to quantum annealing is the concept of an objective function.
- This function encodes the problem we want to solve and is translated into the quantum system's energy landscape.
- The goal is to find the configuration of qubits that minimizes the energy, representing the optimal solution to the problem.

Final State

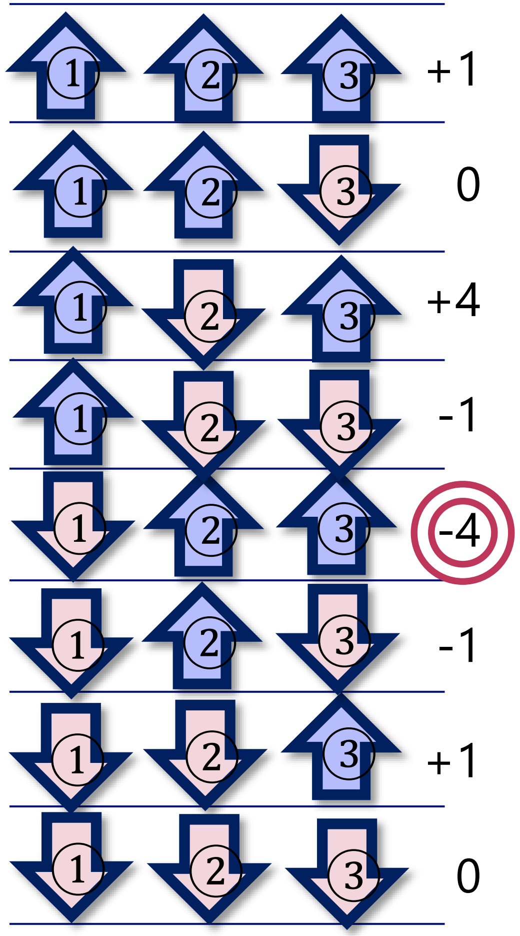
- As the annealing process progresses, the quantum system settles into a final state.
- The configuration of qubits in this state corresponds to the solution of the problem encoded in the objective function.

Annealing

- A variable can be denoted as taking values of either **TRUE (0)** or **FALSE (1)**, *or* denoted as magnetic spins **spin up (↑)** and **spin down (↓)**

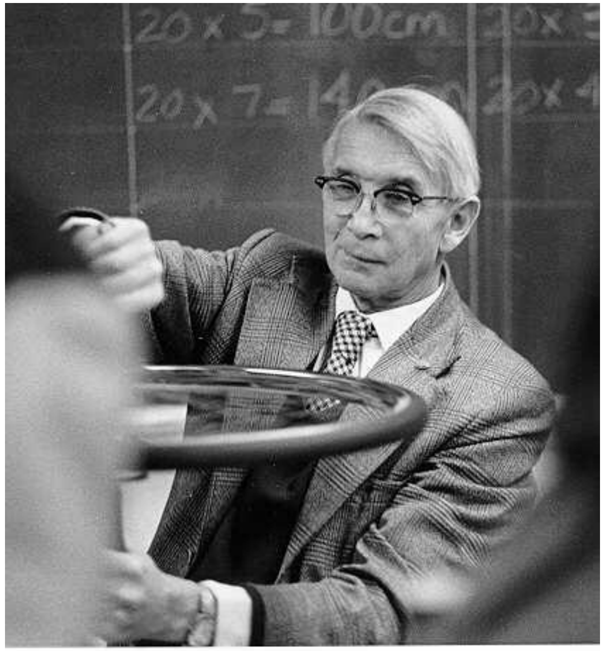
- If the number of spins is 3, the optimal one is obtained from 2 x 2 x 2 = 8 combinations in a single annealing process.
  - If 10 spins, $2^{10}=1024$
  - If 20 spins, $2^{20}≒1$ million
  - If 30 spins, $2^{30}≒1$ billion
  - If 1000 spins, $2^{1000}≒1.071509\times10^{301}$

We can find a good combination out of a huge number of combinations!



What is the Ising model?

◆ Simplified model to calculate the direction of "spin" of atoms for composing a crystal



Dr. E. Ising

(Exhibit) http://theor.jinr.ru/~kuzemsky/isingbio.html

■ Ising Model

- Simplified representation of the behavior of magnetic materials
- Equation for the energy of the entire model (Hamiltonian)

$$H = \sum_{i<j=1}^{N} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{N} h_i\sigma_i$$

- $\sigma_i = \pm 1$: Ising spin (small magnet)
- J_{ij} : The interaction coefficient between the spins i and j (whether the two spins want to point in the same or opposite direction)
- h_i : Bias (magnetic field) on spin i (which way that spin wants to face)

QUBO(Quadratic Unconstrained Binary Optimization)

- Quantum Unconstrained Binary Optimization (QUBO) is a framework used in quantum computing.
- It involves representing optimization problems using binary variables and is particularly suitable for solving combinatorial optimization problems.
- In simpler terms, it's a way of expressing problems in a form that quantum computers can efficiently tackle.

Example:

- You have three tasks (A, B, C), and each task can either be done (1) or not done (0). You want to maximize the total value, where the value of each task is as follows:

  Task A: 3 points

  Task B: 5 points

  Task C: 2 points

- QUBO:
  - Binary variables: x_A , x_B, x_c
  - Objective function: $3x_A + 5x_B + 2x_c$

Let’s Practice QUBO

- Problem: You have three tasks (A, B, C), and each task can either be done (1) or not done (0). You want to maximize the total value, where the value of each task is as follows:

  Task A: 7 points

  Task B: 3 points

  Task C: 8 points

- Solution: $7x_A + 3x_B + 8x_c$

Let's Practice QUBO

- Problem: You have two tasks (X, Y), and each task can either be done (1) or not done (0). Assign binary variables and write the objective function to maximize the total value, where the value of each task is as follows:

  Task X: 4 points

  Task Y: 6 points

- Solution: $4x_X + 6x_Y$

Let's Practice QUBO

- Represent the following constraint using binary variables in a QUBO form:

Constraint: At most two out of three tasks (X, Y, Z) can be selected simultaneously.

- Solution: $E(x) = x_X + x_Y + x_Z - 2\, x_X \cdot x_Y - 2\, x_X \cdot x_Z - 2\, x_Y \cdot x_Z$

Let’s Practice QUBO

- **Traveling Salesman Problem (TSP)**

Given a TSP with four cities (A, B, C, D) and distances between them:

$dAB = 2$

$dAC = 5$

$dAD = 3$

$dBC = 4$

$dBD = 6$

$dCD = 1$

Write the QUBO expression to minimize the total distance traveled.

- Solution:

$$E(x) = 2 \cdot (x_{AB} \cdot x_{BA}) + 5 \cdot (x_{AC} \cdot x_{CA}) + 3 \cdot (x_{AD} \cdot x_{DA}) + 4 \cdot (x_{BC} \cdot x_{CB}) + 6 \cdot (x_{BD} \cdot x_{DB}) + 1 \cdot (x_{CD} \cdot x_{DC})$$

QUBO format important in formulation

- QUBO(Quadratic Unconstrained Binary Optimization)
  - Spin values are expressed as "+1" and "-1" in Ising format.

$$H = \sum_{i<j=1}^{N} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{N} h_i\sigma_i \qquad \sigma_i = \pm 1$$

  - Spin values are expressed as "0" and "1" in QUBO format.

$$H = \sum_{i,j} Q_{ij}x_ix_j \qquad x_i = 0, 1$$

  - The two above can be converted to each other using the following formula.

$$\left(x_i = \frac{1-\sigma_i}{2}\right)$$

- When formulating actual problems, QUBO (0/1) is often easier to think about.

Example of conversion from a problem to QUBO

Problem: To minimize the sum of the numbers Select two boxes

17	21	19

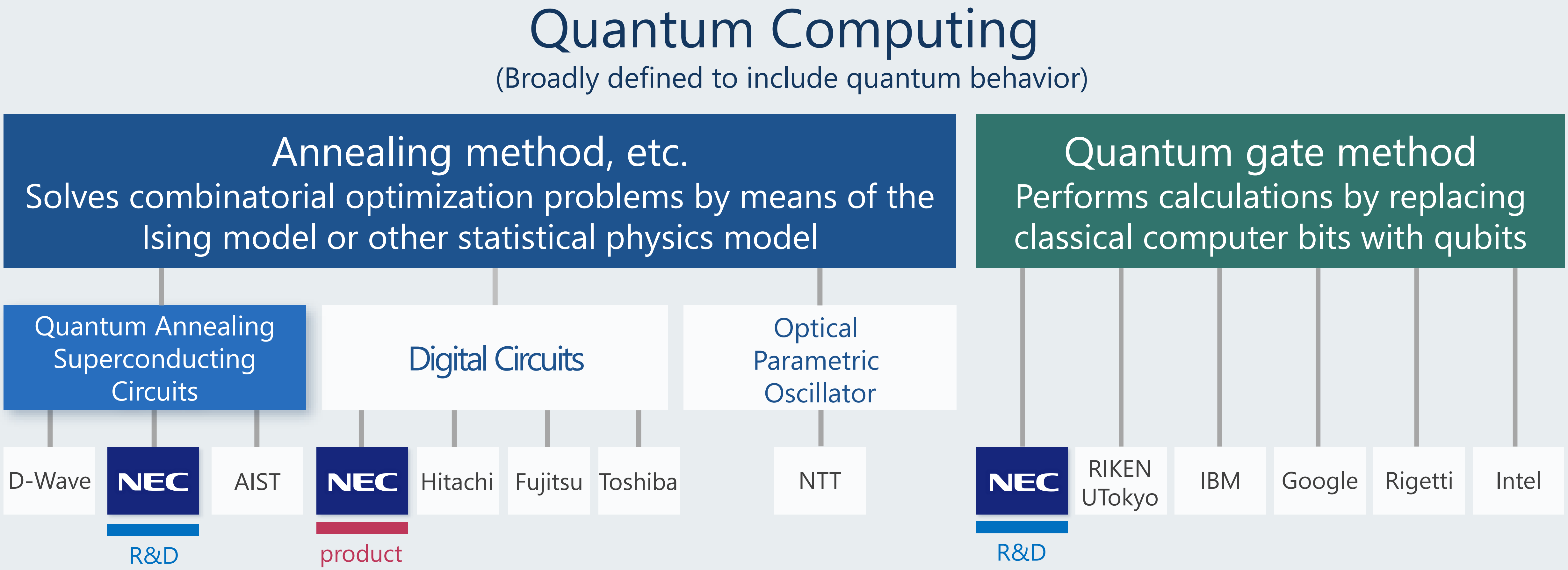
1. Variable definition
 Selecting box i is denoted by xi=1. Conversely, xi=0 indicates that that box i is not selected.
2. Objective function
 Minimize $(17x_1+21x_2+19x_3)$.
3. Creating constraint expressions and converting them to penalty functions
 $x_1+x_2+x_3=2 \rightarrow (x_1+x_2+x_3-2)^2$
4. QUBO
 $(17x_1+21x_2+19x_3)+\gamma(x_1+x_2+x_3-2)^2$
 γ is the weight constant.

If the constraint is not satisfied, this term will be large (not a minimum value)

5. Solving the QUBO
 $x_1=1, x_2=0, x_3=1$
 :

QUBO is the standard model in quantum annealing and is used in many simulated annealing engines

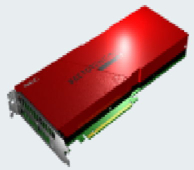
In addition to focusing on the quantum annealing method to address society's optimization needs, NEC is also promoting research and development toward practical application of the gate-based method.



*Based on NEC's survey.　(Due to limited space, not all institutions are covered.)

NEC's Initiatives in Quantum Computing

Since succeeding in the world's first demonstration of solid-state qubit operation, NEC has been working towards the social implementation of quantum computing.



D-Wave The Quantum Computing Company™

2020 to 2021

Vector Annealing on Vector Engine Accelerator

Launched service by an annealing machine for dealing with large-scale problems

2030

Aiming to provide models for solving issues using quantum annealing and quantum gate

1999 | 2020 | 2030 | 2040

1999

World's 1st

Demonstrated solid-state qubit operation

(Published in Nature*1)

2023

Aim to obtain extraordinary long coherence time on quantum annealing machine

*2

2040

Promote MOONSHOT program to achieve practical application of gate-based method

MOONSHOT RESEARCH & DEVELOPMENT PROGRAM

*1: Y. Nakamura et al., Nature 398, 786 (1999)

*2; Based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Solving Social Issues Using Quantum Computing

NEC is trying to apply QC technologies for "**deployable use**"

Development with Co-creation Partners

SMBC Group/ JRI / NEC Platforms / NEC Fielding etc.

Advertisement Infrastructure

- Matching/ Recommendation
- Com. base station
- Surveillance sensor

Manufacturing

- Production plan
- Parts ordering plan

Traffic/Logistics

- Crew shift
- Delivery plan
- Load placement

Financial

- Card fraud detection
- Monte Carlo simulation
- Risk calculation

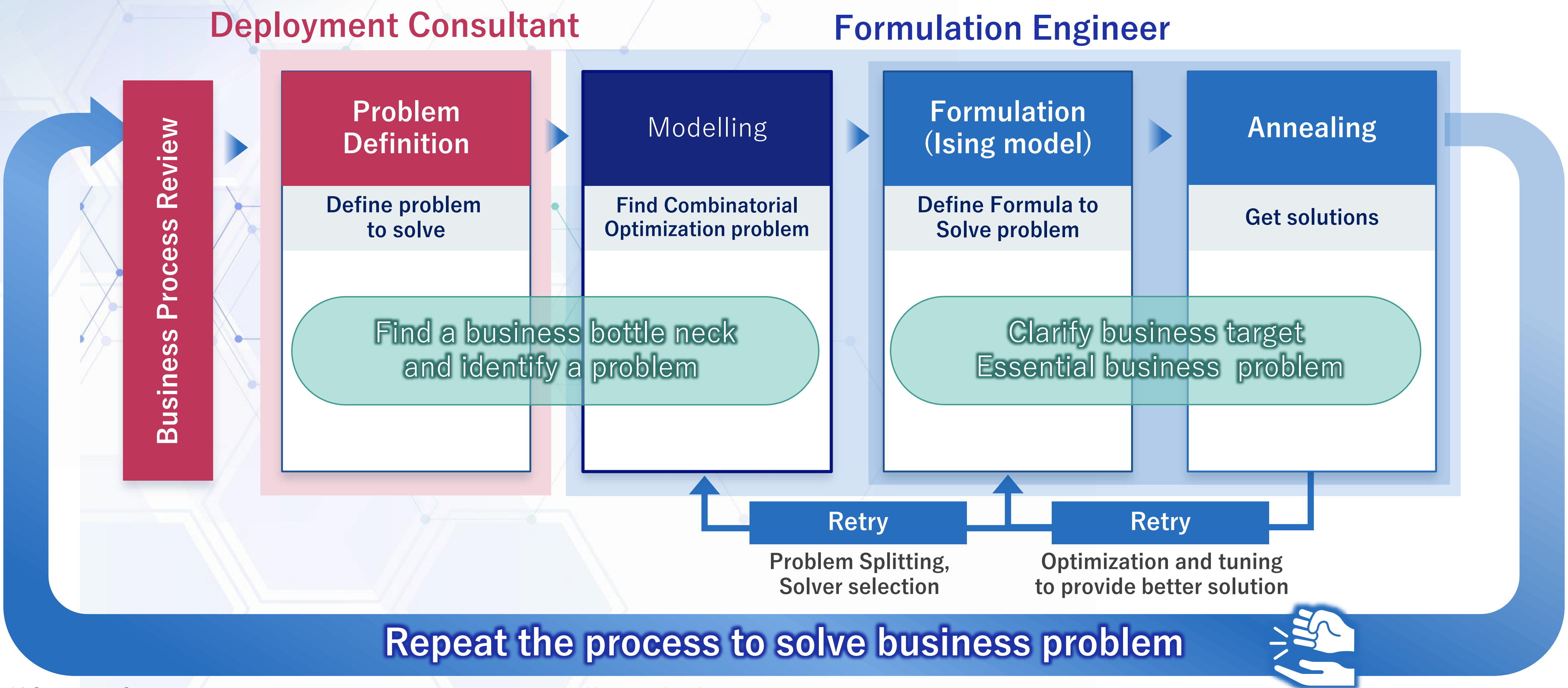
Material/Drug

- Screening
- Experimental
- parameter search

NEC Vector Annealing

Steps to solve a problem with an annealing machine

To apply optimization technique (QA, SA, Mathematical methods, etc.) into real operation in order to improve customer's business processes

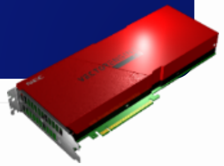


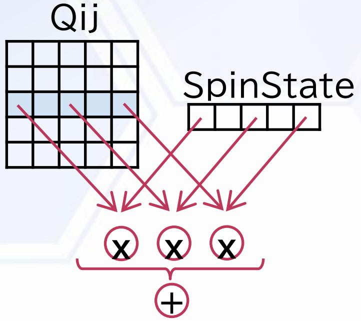
(参考) [1708.01625] Traffic flow optimization using a quantum annealer ： https://arxiv.org/abs/1708.01625

Vector Annealing (VA) on Vector Engine Accelerator (VE)

VA Performance is provided by:
Matrix operation acceleration by VE, large and fast memory, and optimized algorithm for VE

Vector operation on VE
Energy calculation is matrix operation



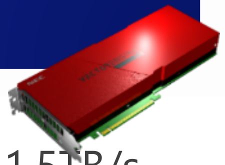



```
for(i=0; i<numSpins; i++)
  DeltaEnergy +=
       Qij[FlipSpin][i] x SpinState[i]
```

Effect from neighbor spin

0 or 1

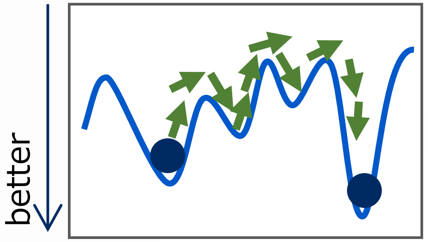
Full connect 100k qbits/VE and high memory bandwidth



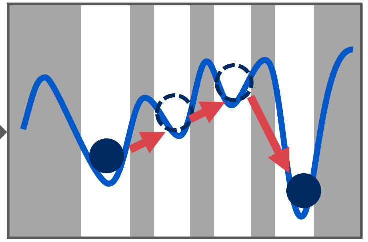
- 48GB memory capacity and 1.5TB/s memory bandwidth
- Multi card supports larger number of qbits (100k qbits x n)$^{1/2}$

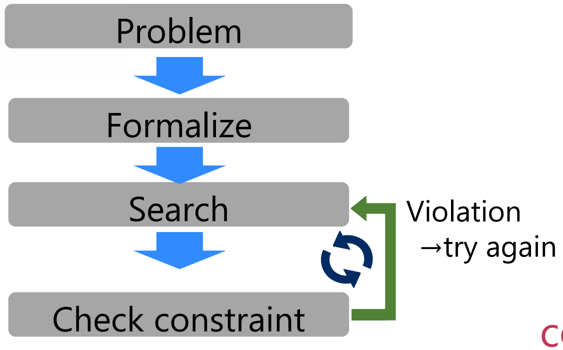
Avoiding Redundant Search and Optimized algorithm for VE

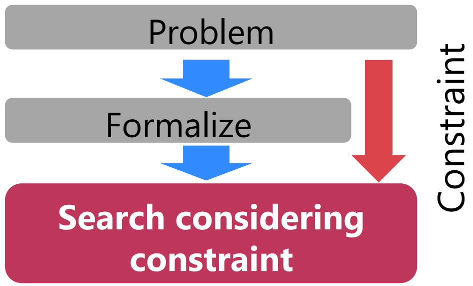
Existing search
Including constraint violations



VA search
skip constraint violations

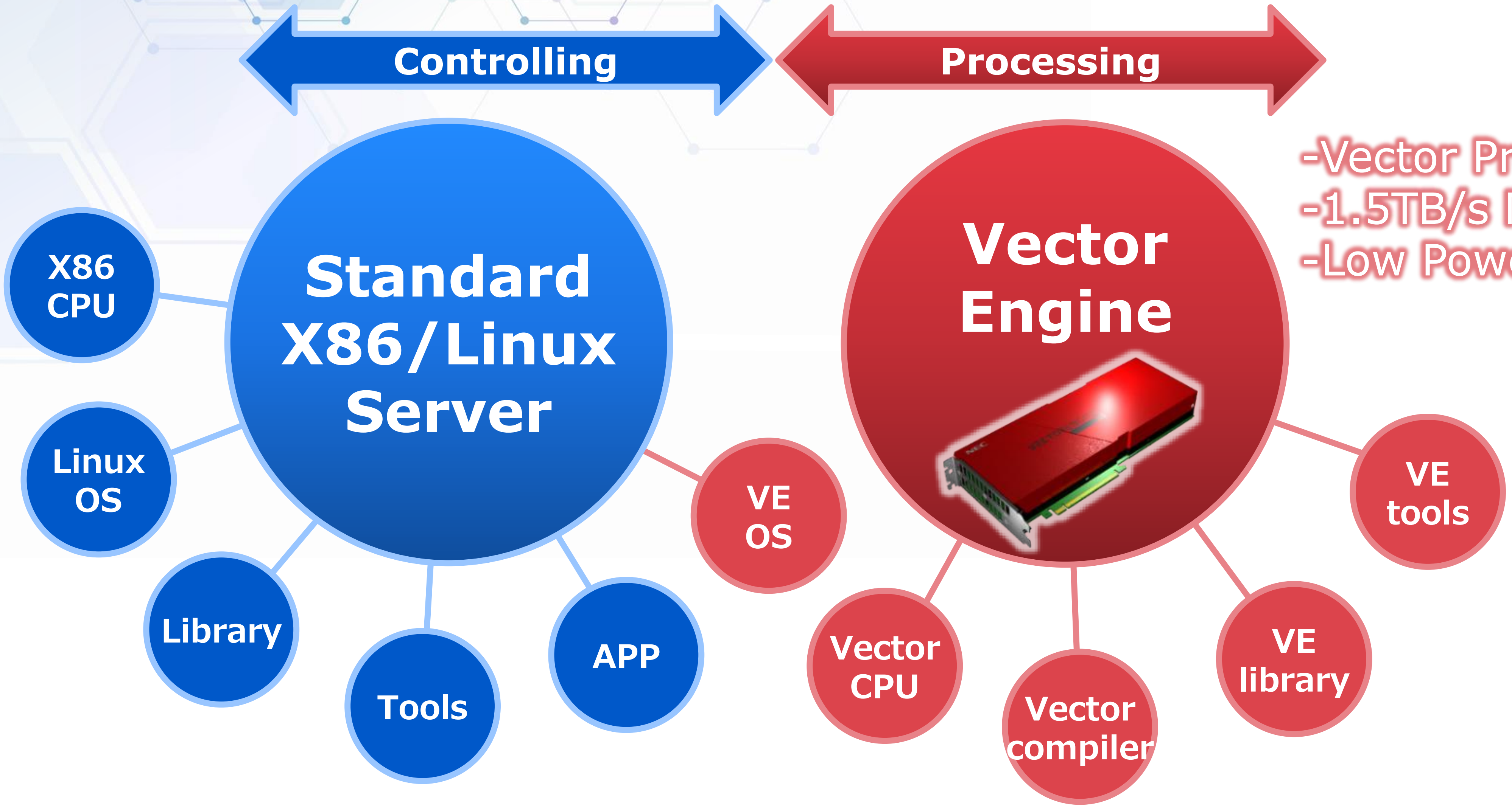






computational complexity reduction

Architecture of SX-Aurora

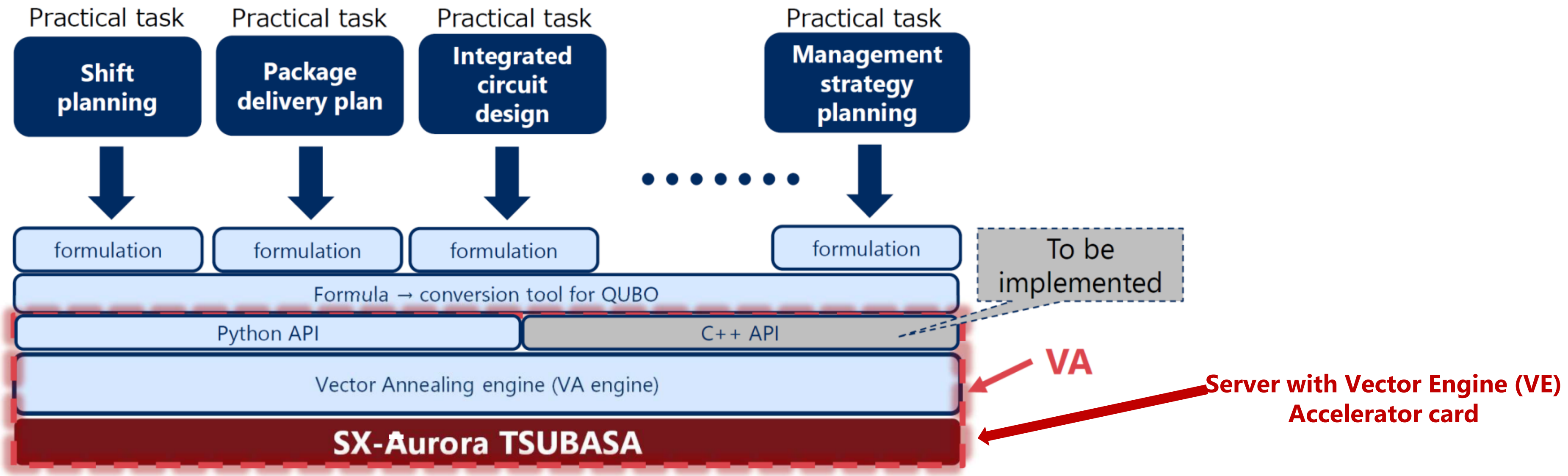


-Vector Processor
-1.5TB/s Memory Access
-Low Power 200W

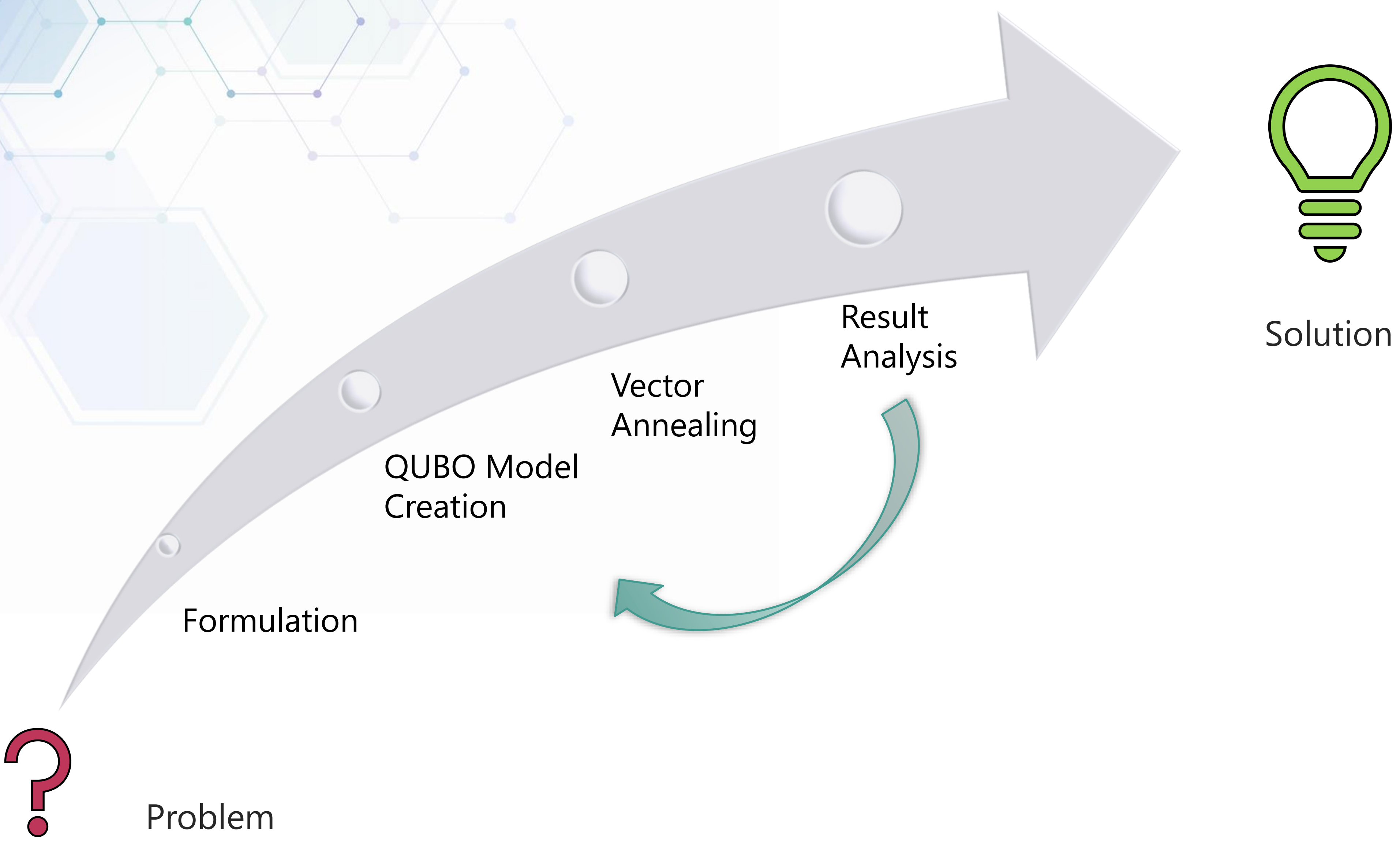
Simulated/Vector Annealing Solution Stack

NEC has developed Vector Engine optimized Simulated Annealing(SA) Engine for solving combinatorial problems.

Input	QUBO format
Problem Size	Up to 300K Qubit/Variables 8 x Vector Engine cards
Connection	32-bit floating point, full connection
Algorithm	Includes our extension to improve result



How to solve problems using Vector Annealing?



Steps from formulation to verification by annealing

◆ Steps

Spin Assignment	manual	Assign number of spins (variables) to solve a combinatorial optimization problem.
Formulation	manual	From the conditions for the optimal combination of spins, determine the equation to be solved by the annealing machine.
Ising/QUBO Model Creation	automatic	From the obtained equation, determine the coefficients of ising model (J_{ij} and h_i).
Annealing	automatic	Verify with an annealing machine.

How “Spin Assignment" and “Formulation" are important to solve for the optimal combination.

Basic steps to solve the problem 1: Spin Assignment

One spin should be assigned to every possible event.

- Spin is assigned to primitive events.
  - Spin = 1 ⇒ The event **has occurred.**
  - Spin 0 ⇒ The event **did not occur.**

Let us look at the Number Partitioning Code!

Defining a Problem and Formulation

- Define a problem which can be expressed in as a binary/spin format.
  - For example: Number partitioning problem: Divide given set of numbers in two sets, such that the sum of the elements of set is equal to other set.
- Define Constraints: The rules and guidelines we need to follow
  - The two sets sum of elements should be equal
- Define Objective function: What to minimize?
  - Minimize the difference in the sum of different sets
- Formulation:
  - For example: *Min F(x) = (12 - 2*(2*x[0] + 4*x[1] + 6*x[2]))**2*

Input: {2, 4, 6}

Output: There are two subsets expected in output:

1. **Subset 1: if x array element value is '1'**
2. **Subset 0 if x array element value is '0'.**

Define three binary qubits as x[0], x[1], x[2]

Sum of Subset 1 or S1 = 2*x[0] + 4*x[1] + 6*x[2]

Sum of Subset 0 or S0 = (2+4+6) – S1

Difference should be minimum or "0" or S1 - S0 = 0

Objective is to minimize the difference. Hence, QUBO formulation will be written as:

(Difference)2 = (12 - 2*(2*x[0] + 4*x[1] + 6*x[2]))**2**

https://arxiv.org/ftp/arxiv/papers/1811/1811.11538.pdf

QUBO Model Creation

- A QUBO problem is defined using an upper-diagonal matrix Q, which is an N x N upper triangular matrix of real weights, and x, a vector of binary variables, as minimizing the function :
  - Q: Quadratic or highest order is 2
  - U: Unconstrained on variables applied
  - B: Binary or {0,1}
  - O: Optimization

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j$$

$$\min_{x \in \{0,1\}^n} x^T Q x.$$

- QUBO is created as a Hamiltonian Expression as explained in example below. +,-,*,/ and square **2 are arithmetic expressions can be used in forming the expressions.

- Example:
  - QUBO model *or Min F(x) = (12 - 2*(2*x[0] + 4*x[1] + 6*x[2]))**2*

- We use “Pyqubo” for QUBO model creation

```
{('x[0]', 'x[0]'): -80.0,
('x[2]', 'x[2]'): -144.0,
('x[2]', 'x[0]'): 96.0,
('x[1]', 'x[1]'): -128.0,
('x[0]', 'x[1]'): 64.0,
('x[2]', 'x[1]'): 192.0}
144.0
```

QUBO Model Output

Annealing

- Provide the QUBO model to the annealing solver
- Provide following basic parameters:
  - Number of reads: Number of initial samples per annealing (by default define as '1')
  - Number of sweeps: Number of sets of iterations to run over the variables of a problem. More sweeps will usually improve the solution (unless it is already at the global min).

- Advanced options are:
  - Flip options
    - Initialization spin options
    - Fixed spin options etc.
  - Beta or temperature range
  - Vector mode

```
#Annealing
va_model = VectorAnnealing.model(qubo, offset)
sa       = VectorAnnealing.sampler()
results = sa.sample(va_model, num_reads=1, num_sweeps=10)
```

Vector Annealing API Parameters

Sampler class parameter specification

- The sa.sample() function which runs the annealing supports the following parameters in addition to num_reads.

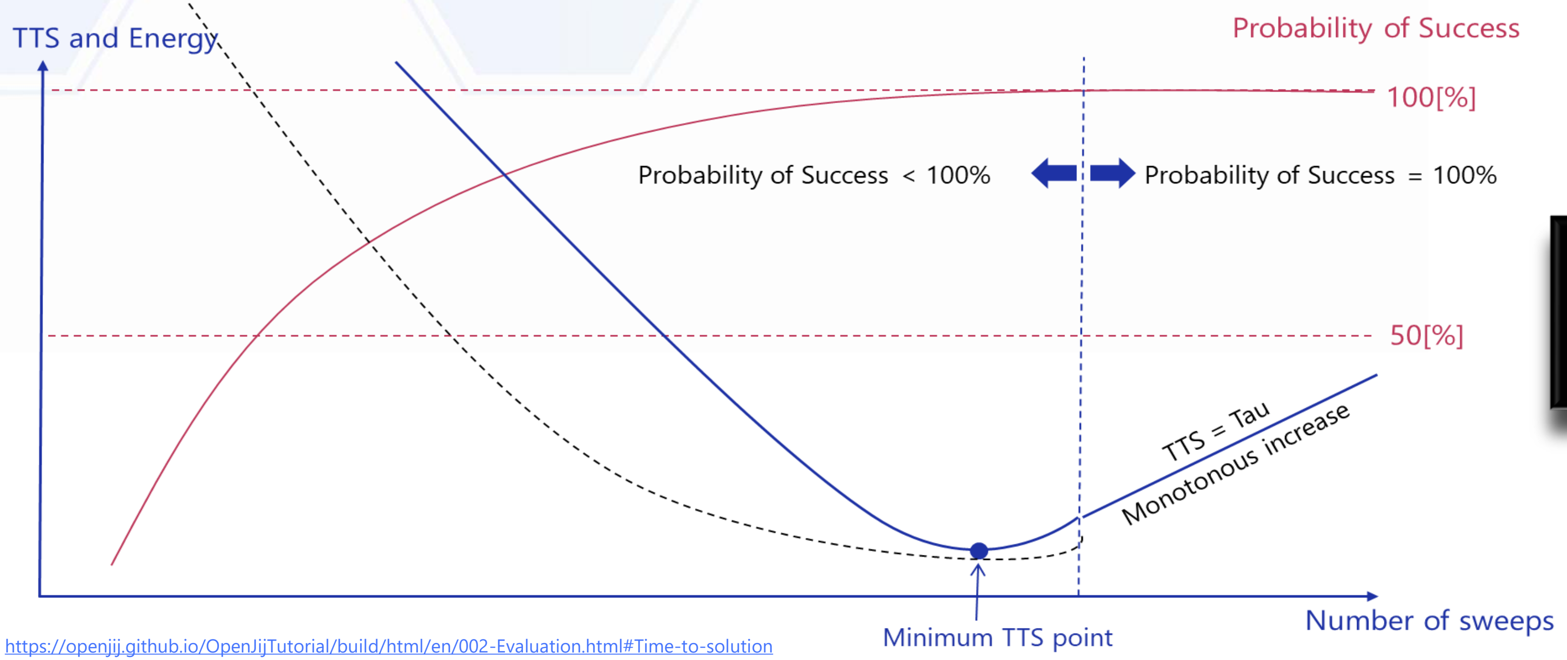
Parameter name	type	default	Description
num_reads	Integer type	1	Number of samplings
num_results	Integer type	None	Number of results. Returns one optimal solution when None or 1 is specified. Returns all annealing results when the same value as with num_reads is specified.
num_sweeps	Integer type	500	Number of annealing sweeps
beta_range	[real number type*,real number type*,integer type]	[10,100,200]	Specify the start and end values of the β parameter which is the inverse of the temperature in [start,end,nsteps]. nsteps is the number of divisions from start to end and can be omitted.
init_spin	Spin array	None	Initial VA spin state specification (For more details, see the "Initial VA spin state specification" section below.)
vector_mode	str type	accuracy	Annealing is run in speed priority when set to speed. Annealing is run in accuracy priority when set to accuracy. Annealing is run in constraint priority when set to constraint. The process will be finished when constraints are met when set to constraint_only.

Result Analysis

- We analyze the following as output:
  - Spin: Output of the annealing
  - Energy: Energy value per solution
  - TTS (Time to Solution): The time required to reach an optimum solution with high probability of success by running multiple annealing processes.

$$\mathrm{TTS}(\tau, p_R) = \tau R = \tau \frac{\ln(1 - p_R)}{\ln\{1 - p_s(\tau)\}}$$

- Solution value against minimum energy with highest probability of success (by performing multiple iterations) is the ideal result.



Output: {'x[0]': 1, 'x[2]': 0, 'x[1]': 1}
energy: 0.0
time: 0.004166000057011843

Result Output showing x[2] of value '6' is group from '0' and x[1] + x[0] = '6' of group '1'

* https://openjij.github.io/OpenJijTutorial/build/html/en/002-Evaluation.html#Time-to-solution

Python code snippet

```
from pyqubo import Array
import VectorAnnealing

#Formulation
x = Array.create('x', shape=3, vartype='BINARY’)
H = (12 - 2*(2*x[0] + 4*x[1] + 6*x[2]))**2

#QUBO Model Creation
model = H.compile()
qubo, offset = model.to_qubo()

#Annealing
va_model = VectorAnnealing.model(qubo, offset)
sa       = VectorAnnealing.sampler()
results = sa.sample(va_model, num_reads=1, num_sweeps=10)

#Result Analysis
print("Output:", results[0].spin)
print("energy:", results[0].energy)
print("time:", results[0].time)
```

Travelling Salesman Problem (TSP)

- Objective Function:
  - Minimize the distance between a pair of locations.

- Constraints:
  - Not to visit the same location more than twice.
  - Not to visit two cities at the same time.

Hd: For all pairs of points, the distance d is set only when moving from point p1 to point p2, and the others are set to 0

$$Hd = \sum_{i=0} \left(\sum_{p1=0, p2 \neq p1} d_{p1p2} x_{i,p1} x_{(i+1),p2} \right)$$

Constraints:

Ha: Not to visit the same location more than twice

$$Ha = strength * \sum_{i=0} \left(\sum_{j=0} x_{i,j} - 1 \right)^2$$

Hb: Not to visit two locations at the same time

$$Hb = strength * \sum_{j=0} \left(\sum_{i=0} x_{i,j} - 1 \right)^2$$

QUBO: ***H* = *Hd* + *Ha* + *Hb***

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j$$

or

$$\min_{x \in \{0,1\}^n} x^T Q x.$$

Flip Option (1/8)

How to use Flip option

- By specifying Flip option, you can efficiently obtain the results in a simulation.

*These options must be included in Hamiltonian's formulation.

- Sample code: Traveling salesman problem

```
# 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)
```

Define constraints on Onehot

Specify the constraint so that only one city becomes 1 at order i and only one order becomes 1 at city j.

Specify x[0][0] = 1 and x[0][*] = x[*][0] = 0 in order to start from the city at j= 0

Define constraints on Fixed spin

Specify the constraint of Flip options when creating model of Vector Annealing.

Flip Option (2/8)

How to specify Flip option

- One hot constraint
  - The constraint that one of the spin states is “1” in the specified group of spin

A group of One hot constraints

```
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)
```

Define multiple one hot constraints

Specify the defined one hot condition in the variable of onehot

Flip Option (3/8)

How to specify Flip option

- Fixed spin constraint

*Only this option does not need to be included in Hamiltonian's formulation.

  - The constraint that specifies the state of spin to the specified value (0/1)

Specify set of the name of spin and the state of spin(0/1)

```
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)
```

Specify the status of multiple spin

Flip Option (4/8)

How to specify Flip option

- And zero constraint
  - The constraint that at least one of the spin states is “0” in the specified group of spin

A Group of And zero constraint

```
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)
```

Define multiple and zero constraint

Specify the defined and zero condition in the variable of andzero

Flip Option (5/8)

How to specify Flip option

- Or one constraint
  - The constraint that at least one of the spin states is “1” in the specified group of spin

A Group of Or one constraint

```
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)
```

Define multiple or one constraint

Specify the defined or one condition in the variable of orone

Flip Option (6/8)

How to specify Flip option

- Cubic supplement constraint
  - The constraint that spin(x_1, x_2, y_1) always has a value that satisfies expression $y_1 = x_1 x_2$

Describe the constraints of cubic supplement in the order of y_1, x_1, x_2

```
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)
```

Define multiple cubic supplement constraint

Specify the defined cubic supplement condition in the variable of spl

***Need the following Hamiltonian that satisfied expression $y_1 = x_1 x_2$**

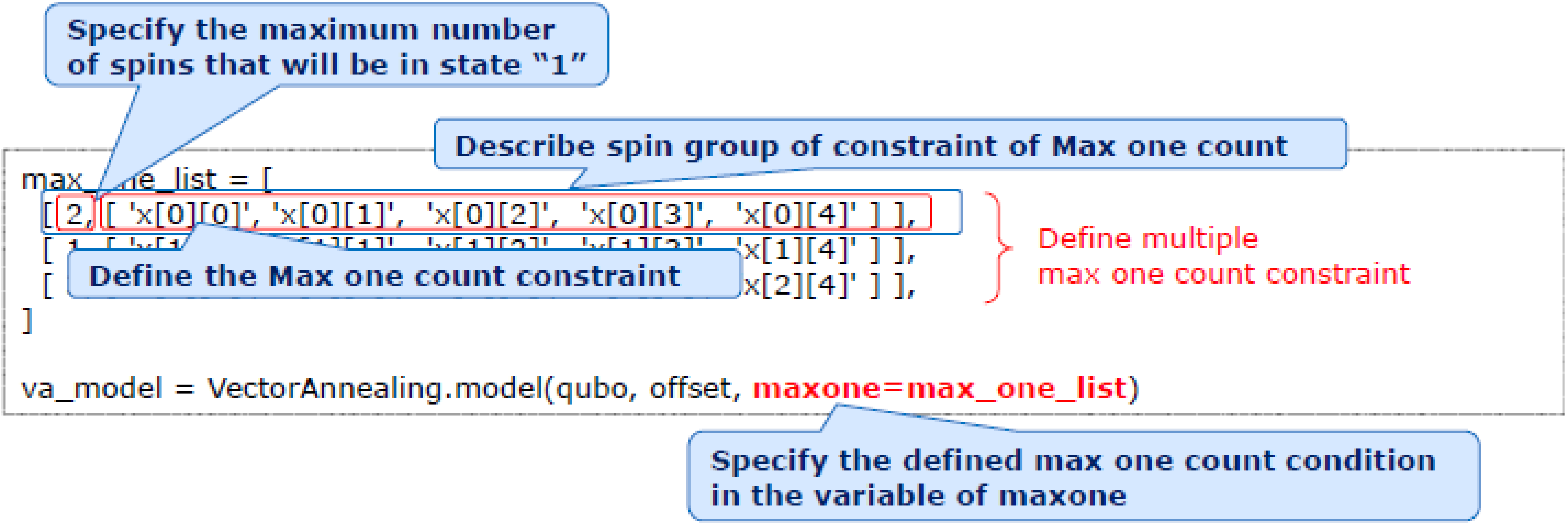
Hamiltonian formulation sample)

```
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]
```

Flip Option (7/8)

How to specify Flip option

- Max one count constraint
  - The constraint that the number of spins having the “1” state in the specified group is equal to or less than the specified number.



Flip Option (8/8)

How to specify Flip option

- Min max one constraint
  - The constraint that the number of spins having the "1" state in the specified group is equal to the specified range.
    - The range parameter is defined as [MIN, MAX, [spin0, spin1, spin2, ...]]
      - In the case of MIN =< MAX
        MIN =< the number of spins having the "1" state =< MAX
      - In the case of MIN > MAX
        the number of spins having the "1" state =< MAX or MIN =< the number of spins having the "1" state

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

Specify MIN=1 (Minimum threshold)

Describe spin group of constraint of Min max one

Specify MAX=3 (Maximum threshold)

Define the Min max one constraint

the number of spins having the "1" state =< MAX(=1) or MIN(=3) =< the number of spins having the "1" state

Define multiple min max one constraint

Let us look at the TSP Code!

Sudoku* Game

- 9 x 9 Squares Number placing problem
- Objective: Is to have one unique number in each box of the matrix and to minimize repetition.
- Rules or Constraints:
  - Numbers from 1 to 9 that do not overlap in a horizontal row.
  - Numbers from 1 to 9 that do not overlap in a vertical columns.
  - Each 3x3 square contains non-overlapping numbers 1-9
- Difficulty Levels: (Depending on hints and numbers to be guessed)
  - Easy
  - Medium
  - Hard

8					5	1		
		1				8		
	4		2				9	
				3				2
1	2	3	4		6	7	8	9
6				1				
	8				9		5	
		2				4		
		7	6					1

Puzzle Example Source: https://www.nikoli.co.jp/en/puzzles/sudoku.html

*Sudoku name is a number placement puzzle which is a copyright of NIKOLI Co., Ltd. Japan

Sudoku game problem definition and formulation

- **Objective Function:**
  - There are 9x9 cells where each cell can take numbers from 1 to 9, making 9x9x9 = 729 cells total that can be expressed in 0 or 1 to represent each number which will fit into the solution space.
    - Where i is the row number from 1 to 9
    - j is the column number from 1 to 9
    - k is the number of each cell, from 1 to 9 also
    - Each X_{ijk} can have value of 0 or 1.

- **Constraints**:
  - There will be 5 constraints from the rule of the puzzle:
    - A single cell can only have 1 number

$$\sum_{k=1}^{9} X_{ijk} = 1\,, \quad \forall ij \in cell$$

    - Each column-j cannot have any duplicate number

$$s.t.\ \sum_{i} X_{ijk} = 1\,, \quad \forall j \in column\,,\ \forall k \in K\ (K = \{1..9\})$$

    - Each row-i cannot have any duplicate number

$$s.t.\ \sum_{j} X_{ijk} = 1\,, \quad \forall i \in row\,,\ \forall k \in K\ (K = \{1..9\})$$

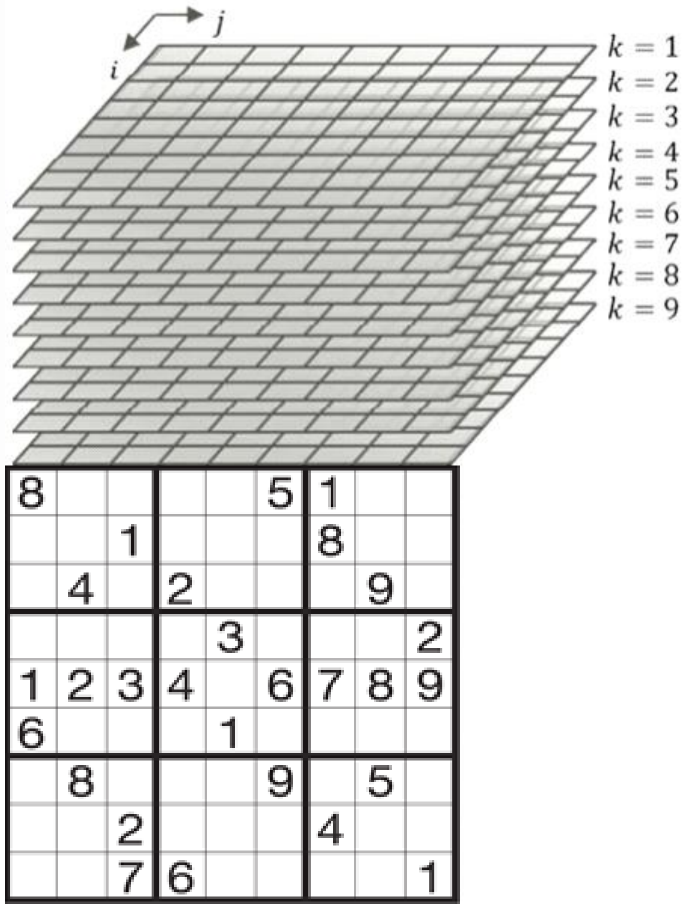
    - Each of the nine 3x3 subgrids cannot have any duplicate number

$$s.t.\ \sum_{j}^{2} \sum_{i}^{2} x_{(i+\upsilon)(j+v)k} = 1\,, \quad \forall k \in K\ (K = \{1..9\})$$

$$\upsilon, v \in \{0,3,6\} : \ \textit{offset to each grids}$$

    - Initial numbers given as “hint” cannot be changed

$$s.t.\ \sum_{hint} X_{ijk} = 1$$



Puzzle Example Source: https://www.nikoli.co.jp/en/puzzles/sudoku.html

*Sudoku name is a number placement puzzle which is a copyright of NIKOLI Co., Ltd. Japan

Let us look at the Sudoku Code!

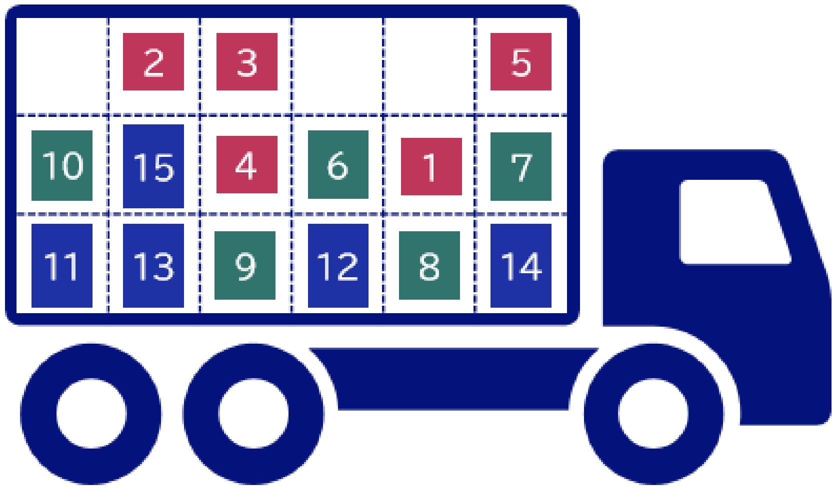
Use Case: Load-Limited Route Optimization Problem

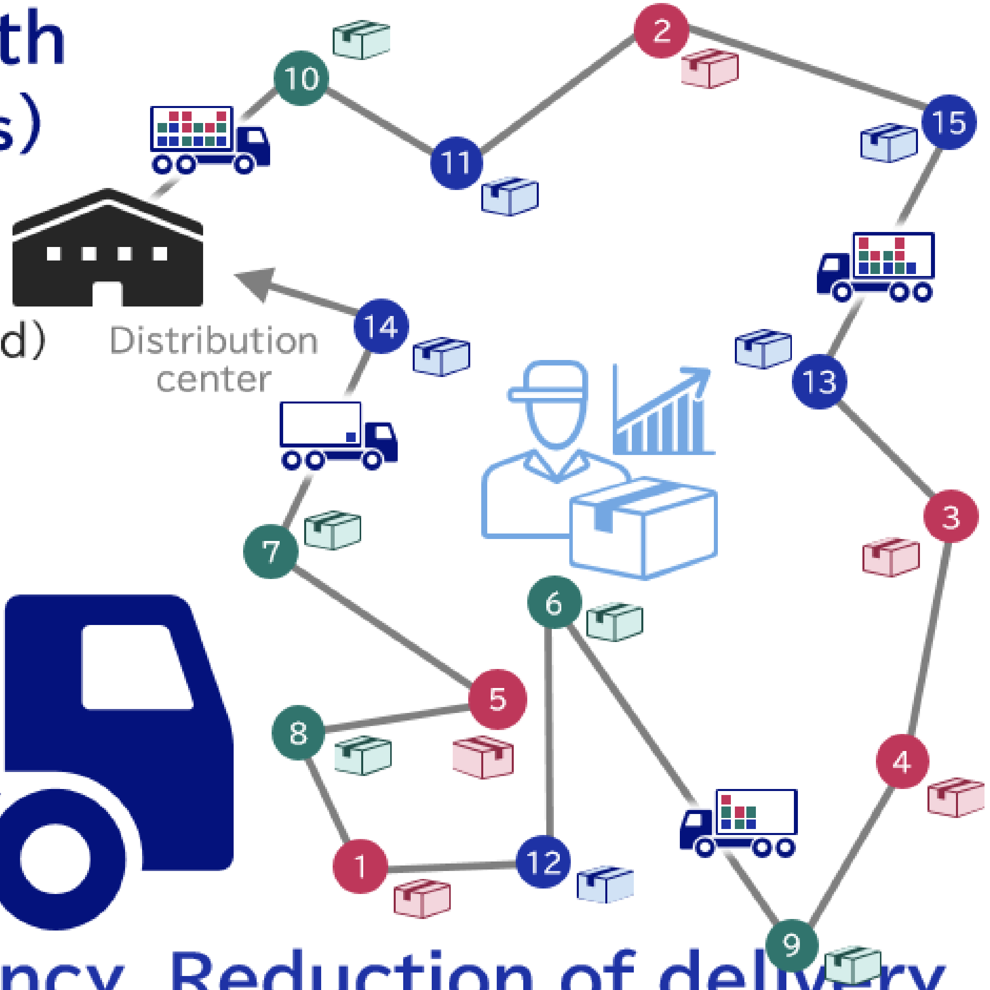
Route optimization problem by taking truck loading restrictions into account

Example: Route optimization problem with special constraints (e.g., loading restrictions)

- Capacity is 18 in 3 rows of 6
  (Cargo is removed from behind and above)
- 15 delivery locations (Allow free space in the cargo bed)
- Heavier cargo cannot be stacked on top

weight	quantity					
5kg	5	1	2	3	4	5
15kg	5	6	7	8	9	10
25kg	5	11	12	13	14	15





Effectiveness Improved delivery efficiency. Reduction of delivery time Reduced burden on delivery staff

Defining Objective Function (Hd)

◆ Minimize the distance of the delivery route.

$$H_d = \underbrace{\sum_{p \in P} d_{C,p} x_{0,p}}_{\text{Distance from the center}} + \underbrace{\sum_{i=0}^{capa-2} \sum_{p_1,p_2 \in Q} d_{p_1,p_2} x_{i,p_1} x_{i+1,p_2}}_{\text{Add up the distance between i and the i+1st delivery destination}} + \underbrace{\sum_{p \in P} d_{p,C} x_{capa-1,p}}_{\text{Distance back to the center}}$$

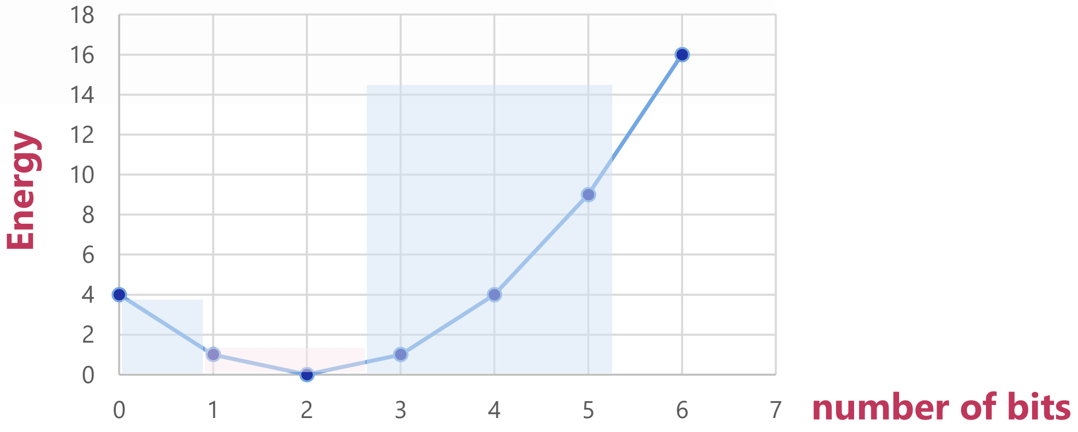
dp1,p2 is the distance between points p1 and p2

Defining Constraints (H1)

- Visit a certain location twice or even three times because there are three extra available boxes (for example: 18 boxes for 15 locations).
  Therefore, the Hamiltonian is created so that <u>the energy is lower if the order bit stands 1 to 3 times for each location.</u>

$$H_1 = \sum_{p \in P} \left(\sum_{i=0}^{capa-1} x_{i,p} - 2 \right)^2$$

If the order bit is high twice, the energy is 0.
If the order bit high once or three times, the energy is 1.
Otherwise, the energy increases exponentially as you move away from 2.



Defining Constraints (H2)

- Since multiple packages cannot be delivered to the same point, only <u>one location can be visited at a time.</u>

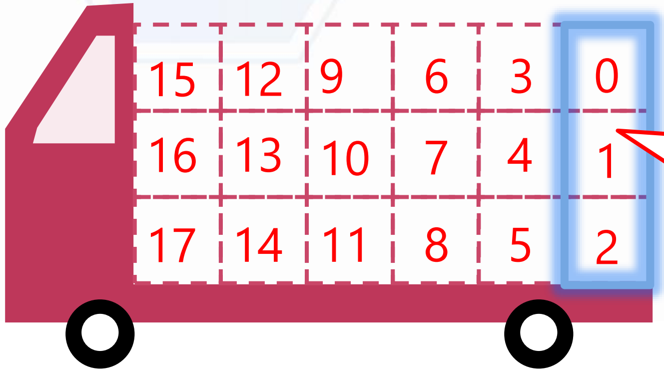
$$H_2 = \sum_{i=0}^{capa-1} \left(\sum_{p \in P} x_{i,p} - 1\right)^2$$

Defining Constraints (H3)

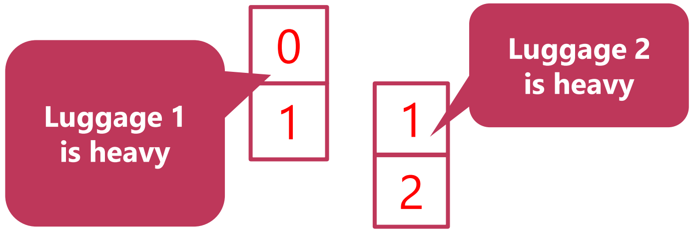
◆ Penalize when the heaviest luggage comes on top.

$$H_3 = \sum_{s_1, s_2 \in space\_pair} \sum_{p_1, p_2 \in R} x_{s_1,p_1} x_{s_2,p_2}$$

R: Pair of points p1, p2 where weight(p1) > weight(p2)



Energy is added only if the upper weight is heavier in the following space combinations.



Defining Constraints (H4)

- Locations that need to be accessed twice or three times should be consecutive.

$$H_4 = -1 * \sum_{p \in P} \Big(\sum_{s_1, s_2 \in space\_pair} x_{s_1,p} x_{s_2,p} \Big)^2$$

Final Hamiltonian Expression

$$H = H_d + 90 * H_1 + 150 * H_2 + 80 * H_3 + 50 * H_4$$

Initially, all the weights are kept small, and the weights are adjusted by trial and error, such as making the weights stronger for constraints that are not satisfied.

Let us look at the LLRO Code!

Benchmark Approach

- Since there are no standard existing benchmarks, we propose the following approach:
  - Select a combinatorial optimization problem
  - complex and large problem, formulate in QUBO and provide to all the generic solvers which support same input format with same constraints or other SA/Hybrid/QA based solvers. Problem in focus is as follows:
    - Load Limited Route Optimization

Benchmark Conditions

- Hardware + Software:
  - VA on VE:
    - VE10B PCIe Card (VE)
    - Vector Annealing 2.0 PoC version
  - CPU or x86:
    - Openjij SA: https://tutorial.openjij.org/build/html/en/index.html
    - D-Wave Neal SA: https://docs.ocean.dwavesys.com/projects/neal/en/latest/reference/sampler.html
  - QPU and Leap Hybrid (Cloud):
    - D-Wave Advantage_system6.1 QA
    - D-Wave Ocean pure QPU Solver: https://docs.ocean.dwavesys.com/en/stable/overview/qpu.html
    - D-Wave Leap Hybrid Solver: https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html
  - Language:
    - Python or ipython notebook

  - Input Dataset (This is for trying larger number of locations higher than 15):
    - http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ALL_tsp.tar.gz
    - Please decompress to get the following tsp files for this benchmark:
      - eil51.tsp
      - eil101.tsp
      - pr76.tsp
      - kroA100.tsp
      - kroA150.tsp
      - kroA200.tsp

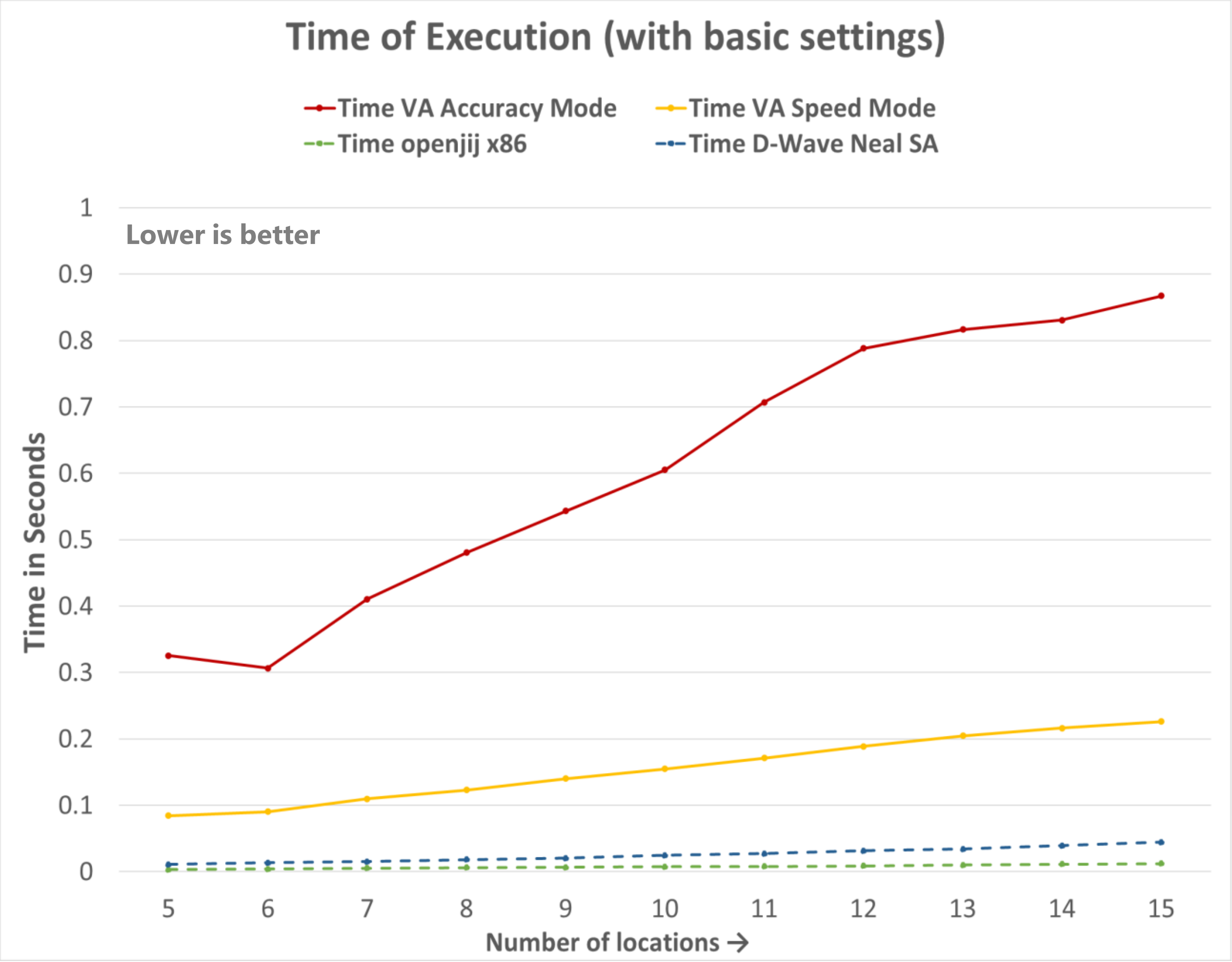
Benchmark Conditions

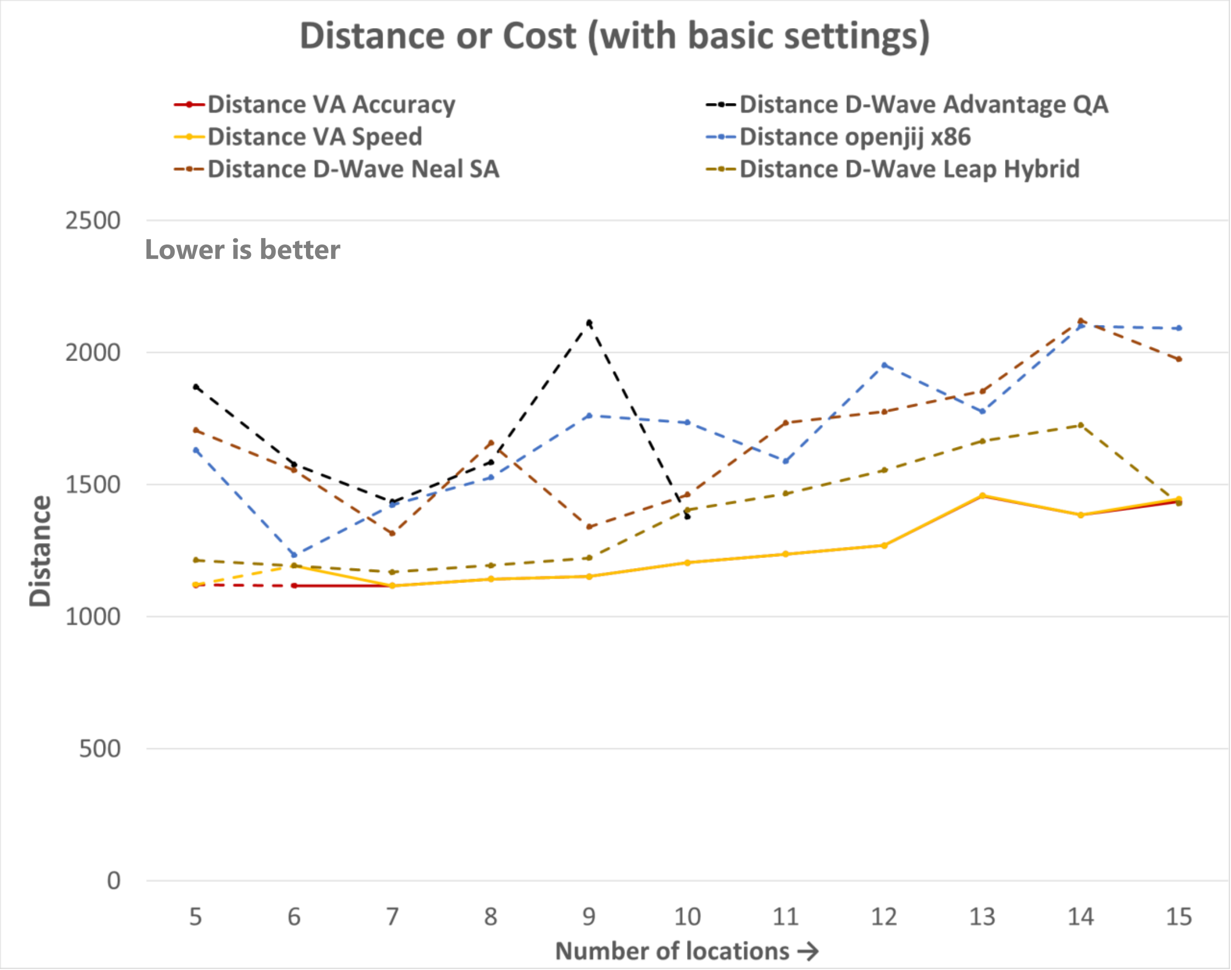
- Problem should be a Combinatorial Optimization Problem.
- Problem Description or objective function and constraints should be same. Every solver is solving the same problem statement.
- The input data should be same.
- The algorithm should be heuristic or meta-heuristic and applying constraints should be possible.
- The output correctness check function should be same.
- Benchmarking Goal:
  - The output should be the best solution quality (with all constraints met).
  - Desirable to have least execution time with the best solution quality.

Benchmark Flow (User Experience):

- **Basic Settings:**
  - **Run with default or basic or minimum parameters or parameter values.**
- Optimization for Best Settings:
  - Increase number of sweeps (VA or SA), increase time limit (Leap Hybrid) and number of Trials (pure QA).
  - Increase number of reads (VA or SA).
  - In-case of constraints not getting met, lower the value of strength of constraints. However, solution quality will degrade.
  - Use TTS and Probability of Success to achieve and get deterministic output every time.
  - Use parameter estimator tool to fine tune the output for a deployable solution.

Result Analysis (15 locations dataset)



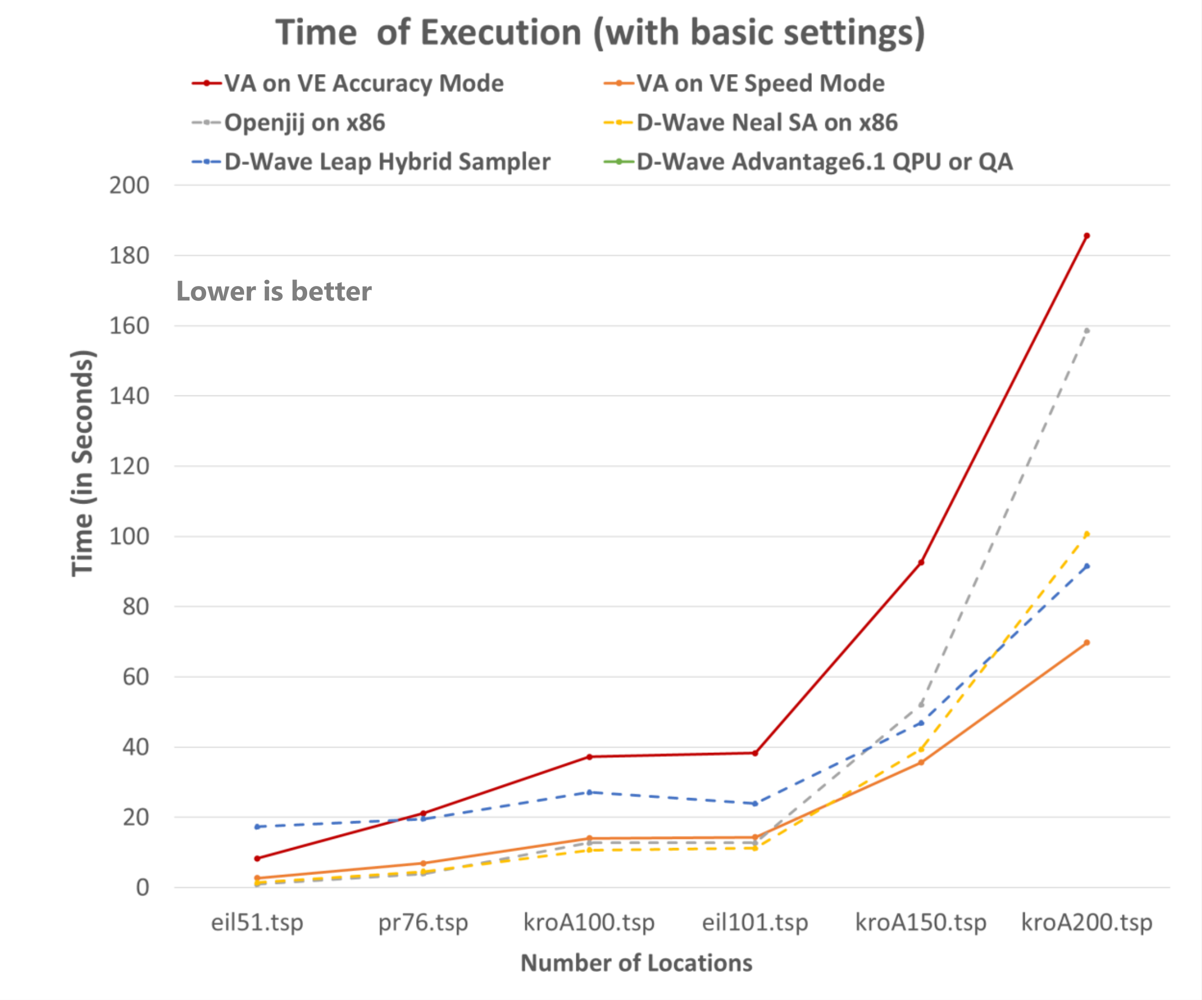


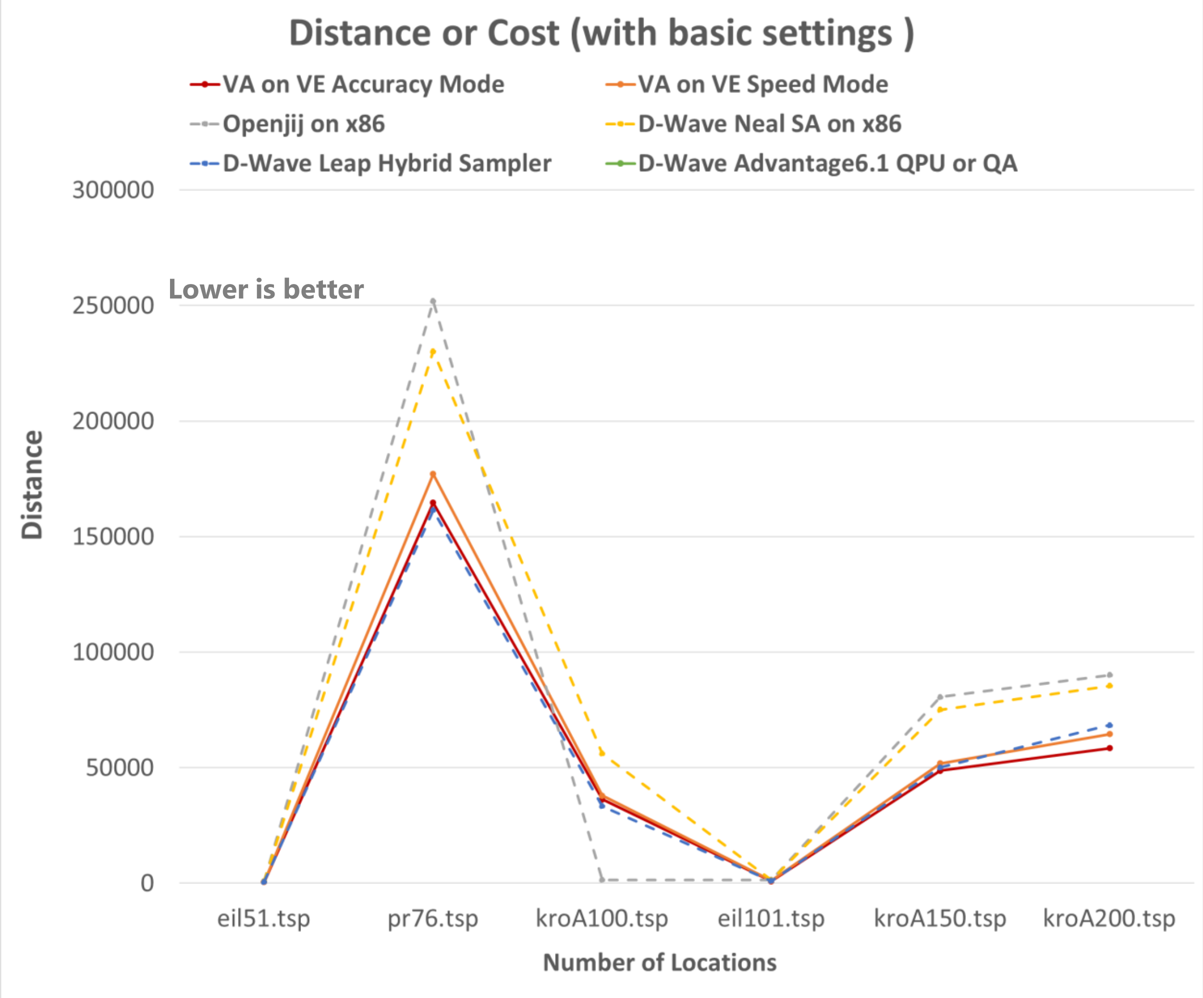
QA and Leap Hybrid Time is not included, because it was cloud access, and it had a large value than 1 second.

- **Dotted Lines represent the constraints were broken by the solver to get the results.**
- **Solid Lines represent no constraints were broken.**

Result Analysis (51 to 200 locations public dataset)

No Hardware embedding found within a time limit due to free cloud access only





- **Dotted Lines represent the constraints were broken by the solver to get the results.**
- **Solid Lines represent no constraints were broken.**

Source of data: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

Benchmark Flow (User Experience):

- Basic Settings:
  - Run with default or basic or minimum parameters or parameter values.

- Optimization for Best Settings:
  - Increase number of sweeps (VA or SA), increase time limit (Leap Hybrid) and number of Trials (pure QA).
  - Increase number of reads (VA or SA).
  - In-case of constraints not getting met, lower the value of strength of constraints. However, solution quality will degrade.
  - Use TTS and Probability of Success to achieve and get deterministic output every time.
  - Use parameter estimator tool to fine tune the output for a deployable solution.

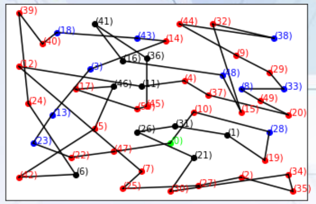
Solution Optimization using number of sweeps

- Goal: To find the best minimum distance as quickly as possible
- Possible Approach specific to the LLRO:
  - Start from the smallest number of sweeps to search for smallest distance in smallest time (number of sweepsαtime of execution of each iteration) even in one iteration.
  - All constraints should be met.
- Results for speed focus (try to use as minimum sweeps as possible):
  - Best Minimum Distance: 501
  - Best Minimum Energy: 62
  - Single Time of Execution: 2.719 seconds
- Results for accuracy focus (try to use as maximum sweeps as possible):
  - Best Minimum Distance: 478
  - Best Minimum Energy: 61
  - Average Time of Execution: 47.837 seconds

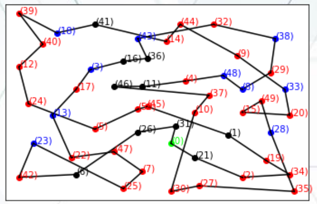
Solutions Exploration

eil51.tsp	Num Trials = 100	VA mode = 'accuracy'								
Number of Sweeps	**Num of Reads**	**TTS (seconds)**	**Best Energy**	**Best Distance**	**Average Energy**	**Average Distance**	**Average Tau (Seconds)**	**Probability of Success**	**Success Parameter**	**Error Type**
10	1	2.427	70	740	72	770	0.173	0.28	No Broken Constraints	Broken Constraints
50	1	1.771	64	589	66	628	0.533	0.75	No Broken Constraints	Broken Constraints
100	1	2.476	63	554	65	596	0.985	0.84	No Broken Constraints	Broken Constraints
200	1	4.495	63	547	63	564	1.919	0.86	No Broken Constraints	Broken Constraints
300	1	5.436	62	524	63	549	2.718	0.9	No Broken Constraints	Broken Constraints
400	1	7.034	62	508	63	541	3.516	0.9	No Broken Constraints	Broken Constraints
500	1	8.63	62	506	62	536	4.315	0.9	No Broken Constraints	Broken Constraints
600	1	7.859	62	501	62	532	5.112	0.95	No Broken Constraints	Broken Constraints
1200	1	18.028	61	497	62	519	9.887	0.92	No Broken Constraints	Broken Constraints
3000	1	24.188	61	494	61	505	24.188	1	No Broken Constraints	
6000	1	163.539	61	478	61	500	47.837	0.74	No Broken Constraints + Minimum energy <= 62 + Minimum distance <= 505	Min Energy or Min Distance or constraints not met
9000	1	178.631	60	480	61	497	71.098	0.84	No Broken Constraints + Minimum energy <= 62 + Minimum distance <= 505	Min Energy or Min Distance or constraints not met
12000	1	204.215	60	478	61	497	94.162	0.86	No Broken Constraints + Minimum energy <= 62 + Minimum distance <= 505	Min Energy or Min Distance or constraints not met
15000	1	264.523	60	482	61	498	117.191	0.87	No Broken Constraints + Minimum energy <= 62 + Minimum distance <= 506	Min Energy or Min Distance or constraints not met

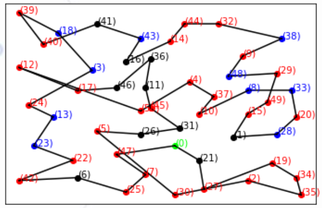
Best Distance Results Output



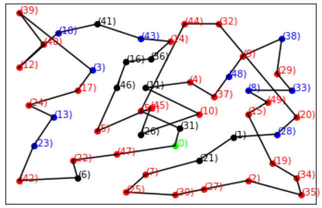
Sweeps = 10 Distance = 740



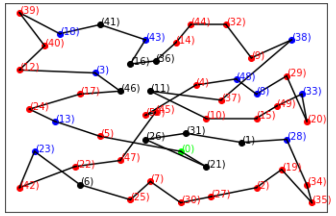
Sweeps = 50 Distance = 589



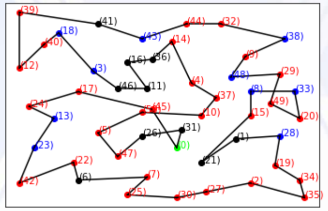
Sweeps = 100 Distance = 554



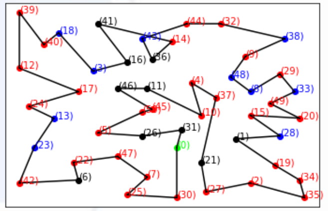
Sweeps = 200 Distance = 547



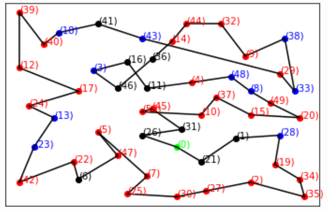
Sweeps = 300 Distance = 524



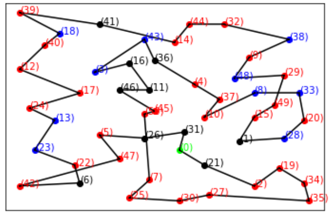
Sweeps = 400 Distance = 508



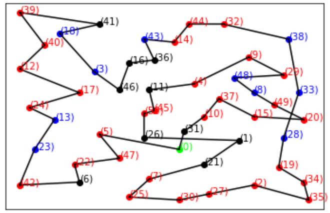
Sweeps = 500 Distance = 506



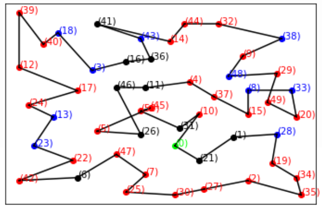
Sweeps = 600 Distance = 501



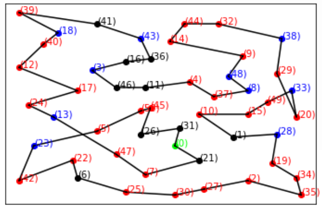
Sweeps = 1200 Distance = 497



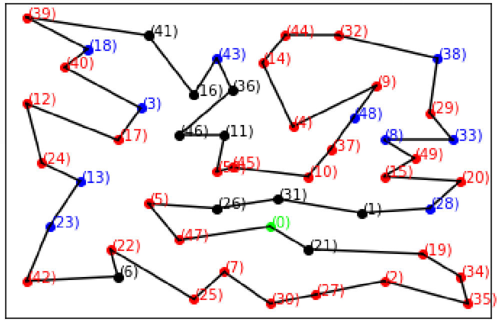
Sweeps = 3000 Distance = 494



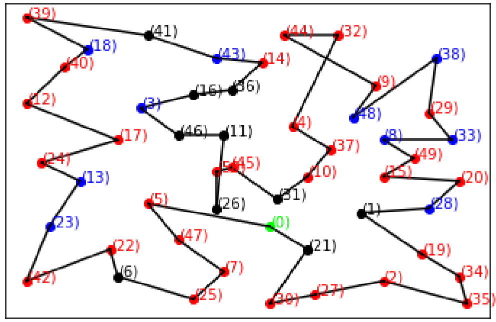
Sweeps = 6000 Distance = 478



Sweeps = 9000 Distance = 480



Sweeps = 12000 Distance = 478



Sweeps = 15000 Distance = 482

Conclusion for Load Limited Route Optimization

- The number of sweeps optimization could reach almost to the optimized solution.
- Further execution time can be optimized using parameter estimator for “beta” values and using less number of sweeps.

Actual Operation

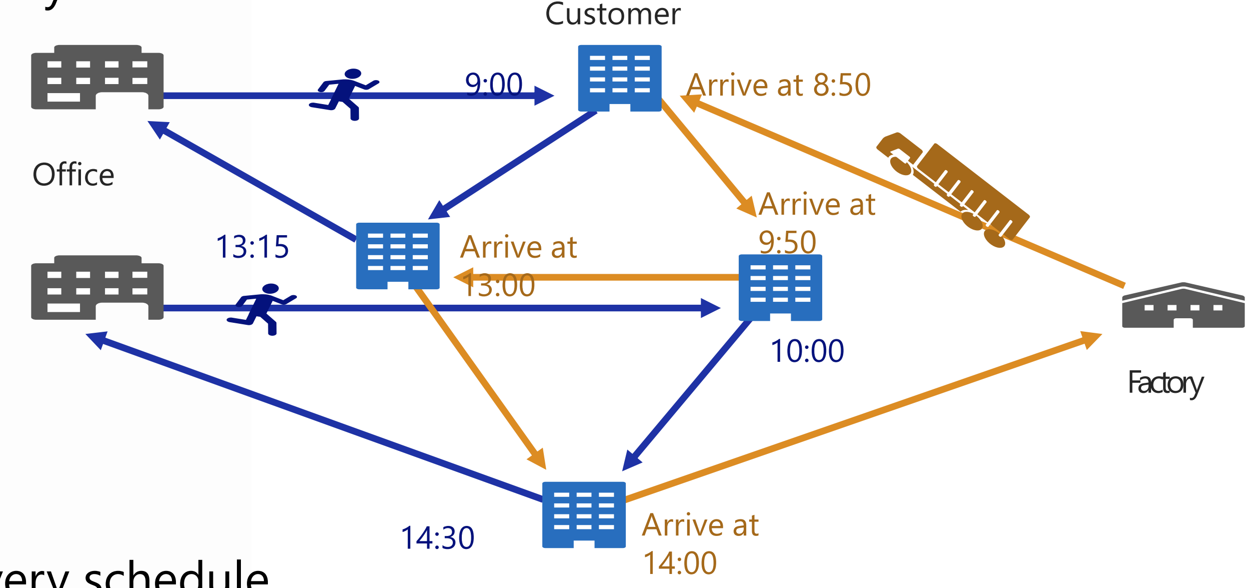
Use Case: **Delivery Route and Schedule Optimization**

for reducing costs, time, energy, CO_2, etc.

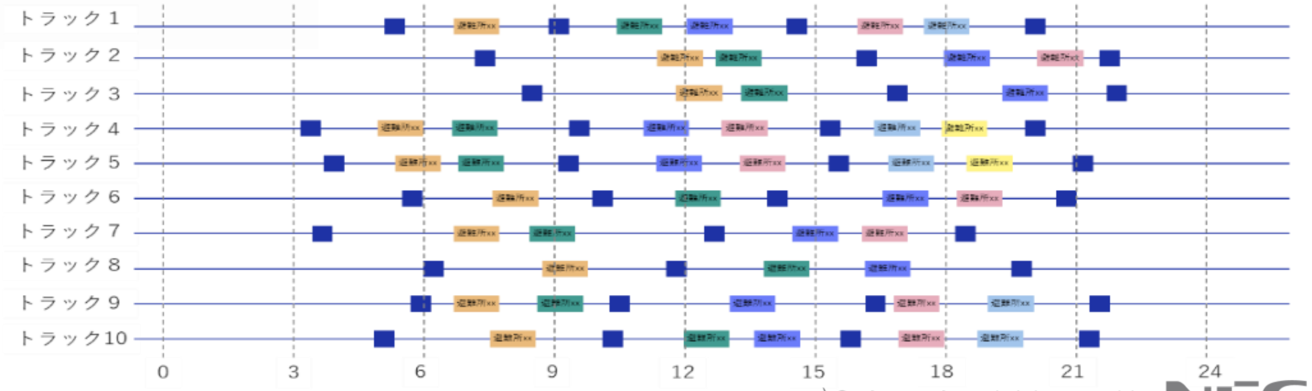
EX. **Delivery of parts and dispatch of Engineers**

- Parts are delivered by truck
- Engineers move by car/train
- Have to consider skills of each engineer

delivery route

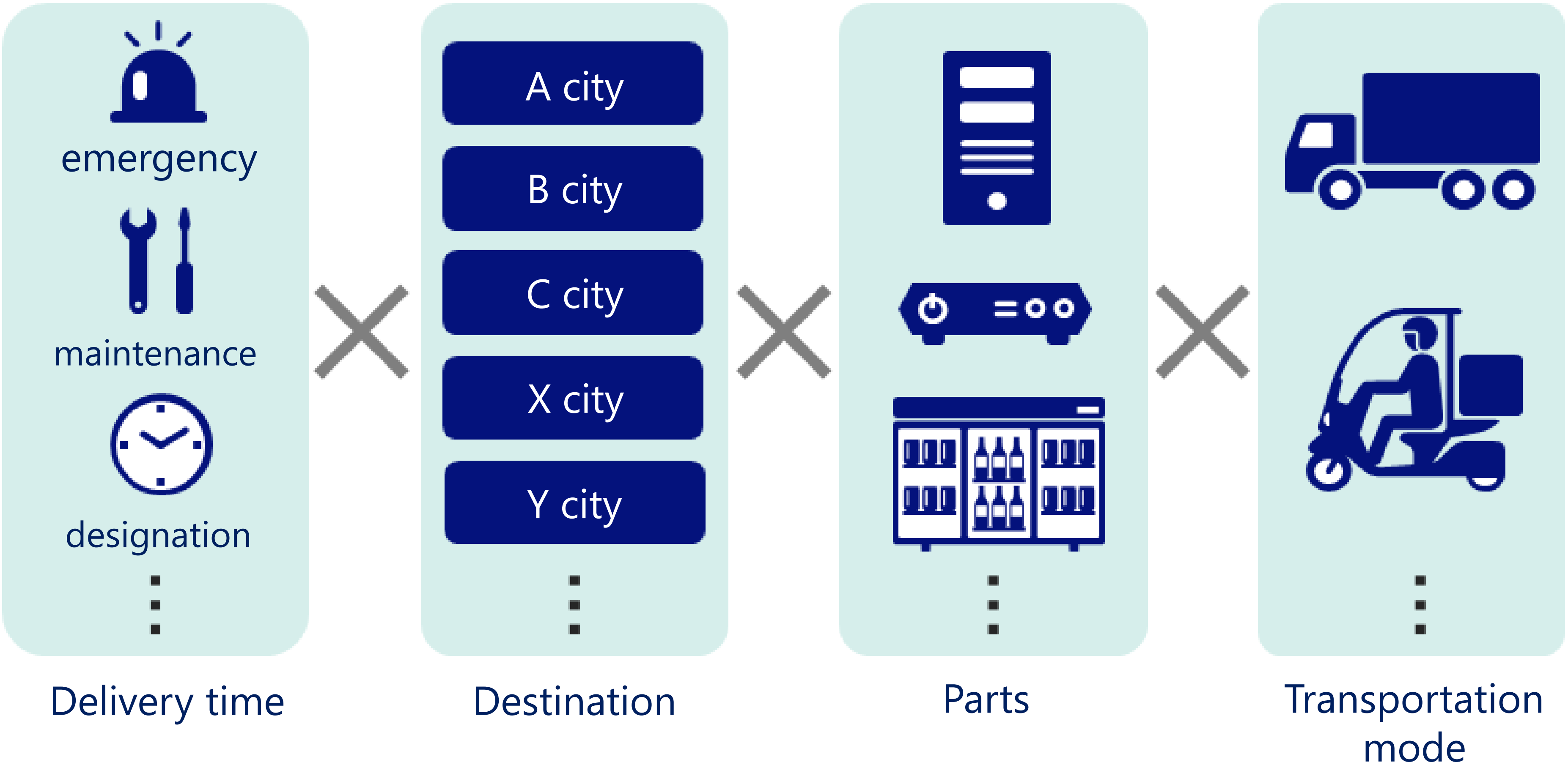


delivery schedule



Delivery Optimization

Combinatorial optimization from huge combination



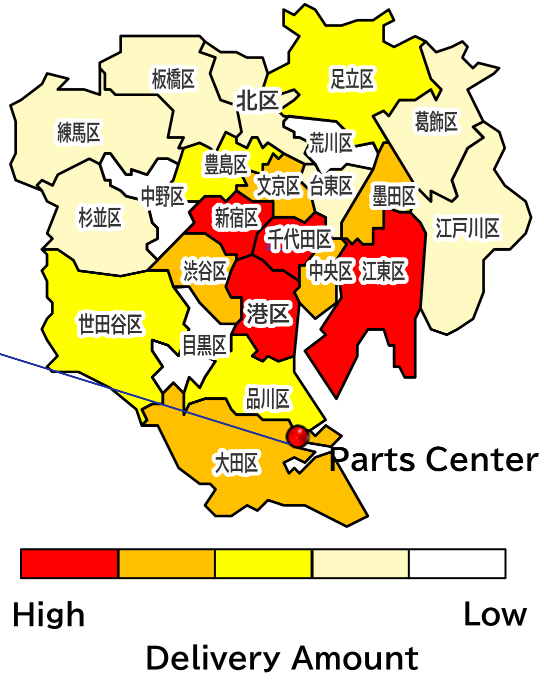
Logistics Problem to be Solved

Tokyo Parts Center

- ☐Warehouse: 6,000m^2, 150k maintenance parts stock
- ☐Delivering to several hundreds destinations in Tokyo by 40 cars

Region :	Tokyo metropolitan area
Operation:	24h x 365days
Delivery Cars :	30 cars and 8 motor bikes
Employee :	43

Tokyo
Parts Center



Delivery Operation

- ☐ Engineers move by public transport
- ☐ Arrival of engineer and each maintenance parts must be same timing
- ☐ Each car/bike brings some parts to deliver some destinations
- ☐ **Huge combination of delivery times, destinations, car/bike, parts**
- ☐ **Professional engineer made delivery plan every day**

Actual Operation by VA as a 1st Step

Start applying VA optimization to delivery order the day before

Actual Operation from October

Delivery order the day before

input

Several 100s Order /day

Professional engineer optimization

120 min./day

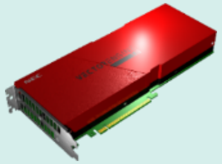
- Delivery Planning
  - ・Delivery time
  - ・Traffic Jam
  - ・Delivery area
  - ・Parts count
  - ・Parts weight
  - ・Transportation mode

Maintenance at customer

Future Plan

Delivery order today & Emergency

30% cost reduction
CO_2 reduction

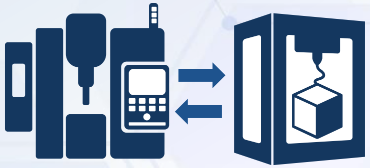


12 min./day

Vector Annealing optimization

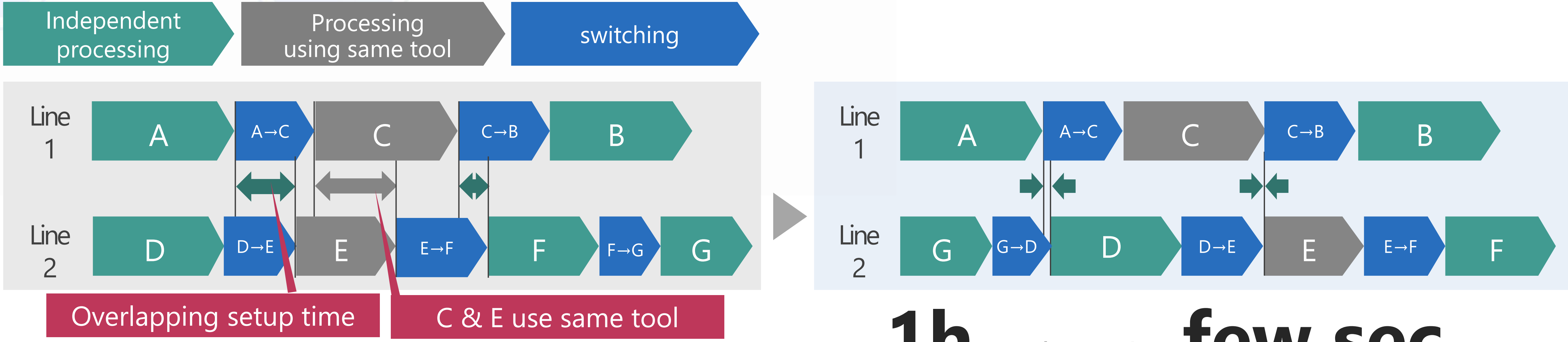
Actual Operation: **Production Planning Optimization**

Optimizing complex planning for multi-product manufacturing lines



Higher versatile processing equipment needs
highly optimized product planning for higher efficiency

- **Switching** products makes idling time of equipment
- Have to consider processing order to avoid **duplication**

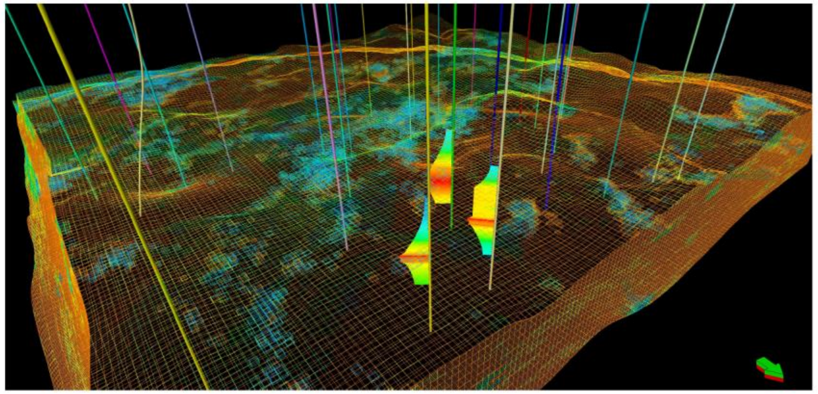


1h accelerated to **few sec.**

\Orchestrating a brighter world NEC

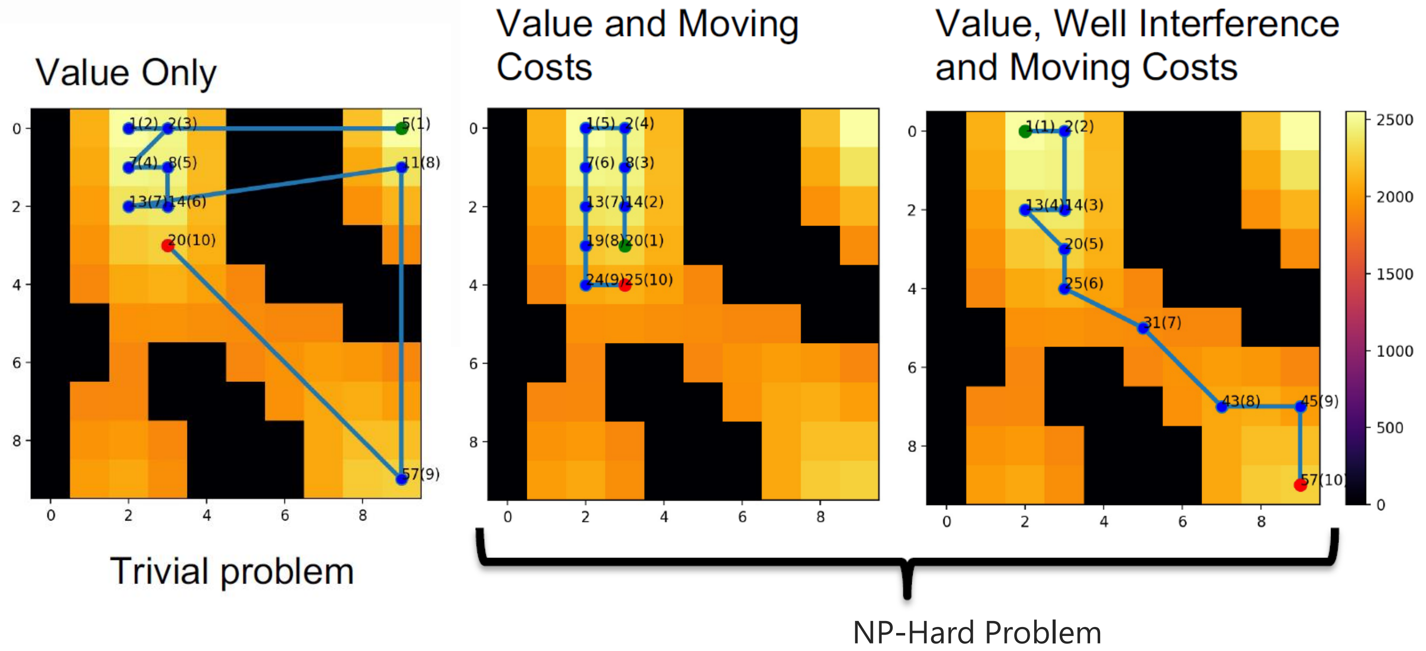
Oil Field Exploration as a Combinatorial Optimization Problem

PoC with oil/gas company



Subsurface modeling is only the beginning of oil field exploration.
Given a map of the distribution of oil and a limited number of resources to develop the field, energy companies must plan a drilling sequence that considers:

- **The value of placing a well at a given location.**
- **The cost of moving a drilling platform from one location to another.**
- **The impact placement of a well has on neighboring locations (well interference)**

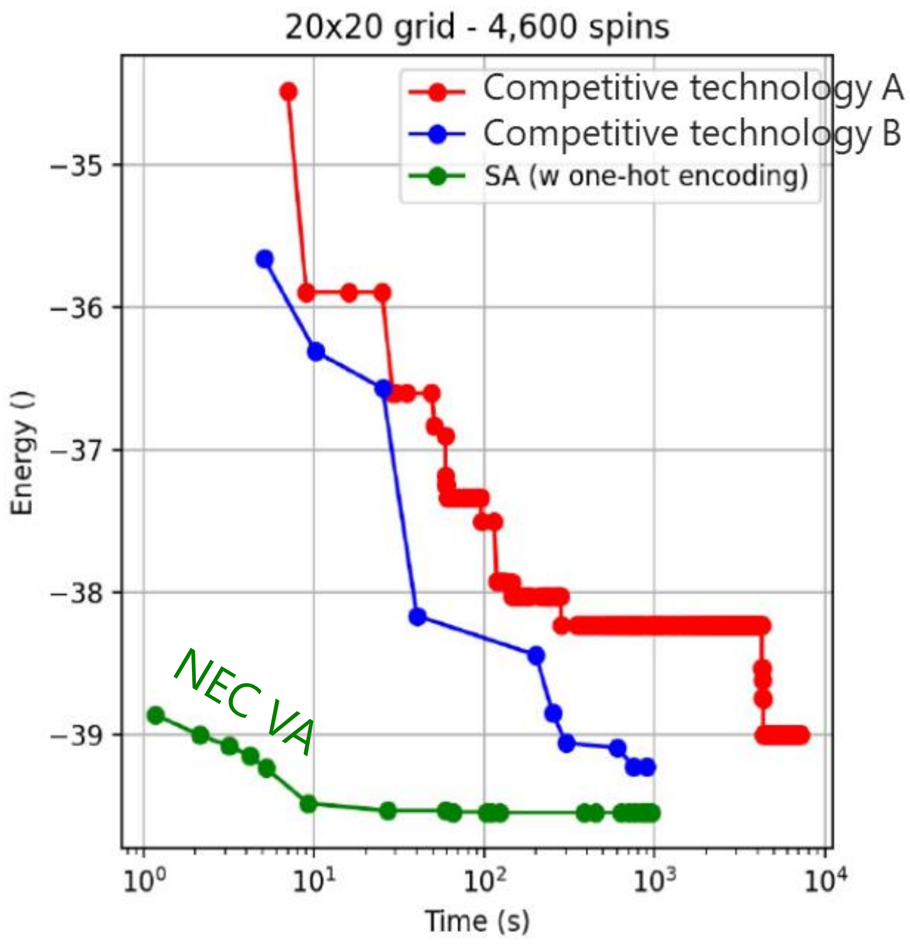


Welling Plan Benchmark

PoC with oil/gas company

- Completed **PoC with software company in US** focusing on energy resource exploration optimization problem.

- NEC VA with external constraints like one-hot encoding provided **best results** in comparison to other ISV SA software running on classical computers as well as accelerators.



NEC achieved lowest energy with shortest time

\Orchestrating a brighter world

NEC