

# Session6



## 交通最適化に関するワークショップ

2023年 2月28日

DEVEL株式会社 比嘉恵一朗

# 目次

今回の量子アニーリング & 古典AIについて

回帰分析とは

実際に体験してみる。

# 今回の量子アニーリング & 古典AIについて

今回は交通最適化問題を量子アニーリングと古典AIを用いて解いてみる。

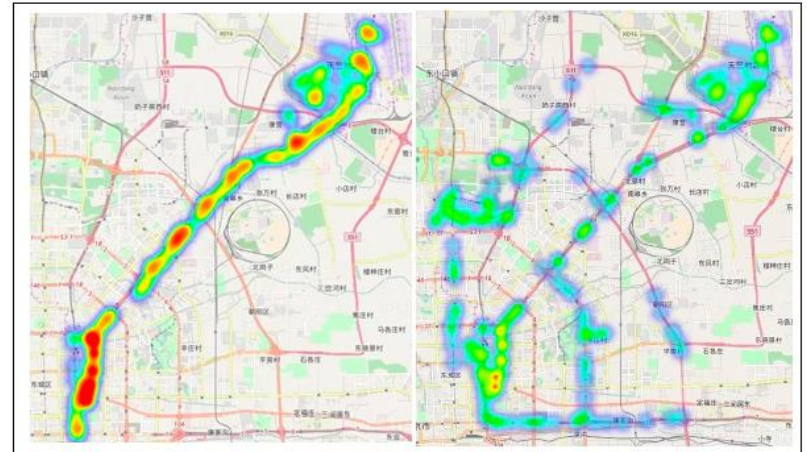
<https://arxiv.org/pdf/1708.01625.pdf>

## 古典AI

曜日と時刻から回帰分析を用いて混雑度を予測する。

## 量子アニーリング

一定以上混雑している場合は別の経路を提案する。



例) 北京の交通最適化

右上の図は北京の空港と市内を行き来する車について最適化を行った結果を示している。

図のヒートマップの部分に車が密集している。→結果からわかる通り渋滞が緩和されている。

# 回帰分析とは

いろいろな数値を比較して行う分析。数値間の関係性や予測、因果関係を調べることに利用できる。

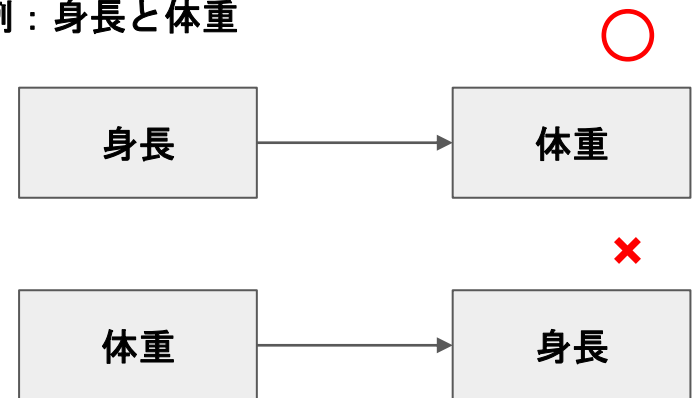
回帰分析では数値に要因と結果の関係を仮定する。つまり以下の図が描ける。



この仮定から以下の2つを分析することができる。

- ・ 要因が結果にどのように影響するか。
- ・ 要因から結果を予測する。

例：身長と体重



仮定が違うと分析できない。

この要因にあたる数値を説明変数、結果にあたる数値を目的変数という。

# 回帰分析とは

別の具体例として、ある地域における時刻と混雑度について考える。

（混雑度とは地域内の経路の本数や通る車の台数などから判断した値。）

基本的に出勤時間の午前中と退勤時間の夕方に混雑すると仮定できる。

この仮定が正しいかを調べる。

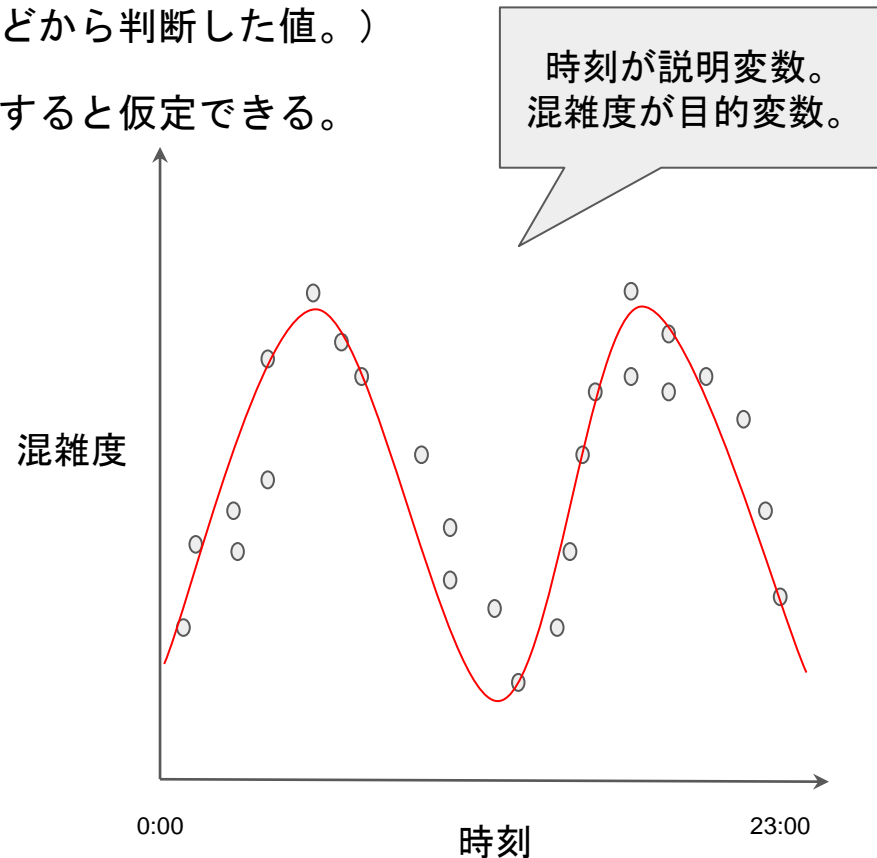


各日にちの時刻と混雑度を右の図のように

プロットすると図の赤い線のように、

関係を示す線を引くことができる。

この線を求めることが回帰分析である。



# 回帰分析の使用例

## 傾向と販売見積もりの評価

説明変数：年齢、教育、経験年数などから、目的変数：販売員の年間総売上を予測する。

## 価格設定の弾力性の分析

説明変数：商品価格から、目的変数：消費が減少するかどうかを確認する。

## 家賃の予測

説明変数：広さ、築年数、駅からの近さなどから、目的変数：家賃の高さを予測する。

## 広告クリエイティブの最適化予測

説明変数：背景（色、模様、質感など）、フォーマット、素材（各種）、企業自身の分析力などから、  
目的変数：広告のクリック率を予測する。

これらの例のように回帰分析では各データ間の関係性を分析することができる。

# 実際に体験してみる。

今回は実問題である交通最適化問題について考えてみる。

## 背景

現在のカーナビシステムでは、基本的に出発地点から目的地点までの最短経路が提案される。この経路は個人にとって最適な経路である。しかし全員が最短経路を選択すると、交通混雑が引き起こされ全体の旅行時間（目的地点までに要する時間）が多くなる可能性がある。

## 目的

交通混雑を緩和したい。

## 課題

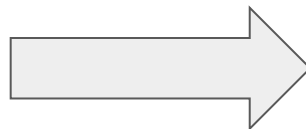
交通混雑を予想して、混雑が見込まれる場合は経路を変更させる。

# 実際に体験してみる。

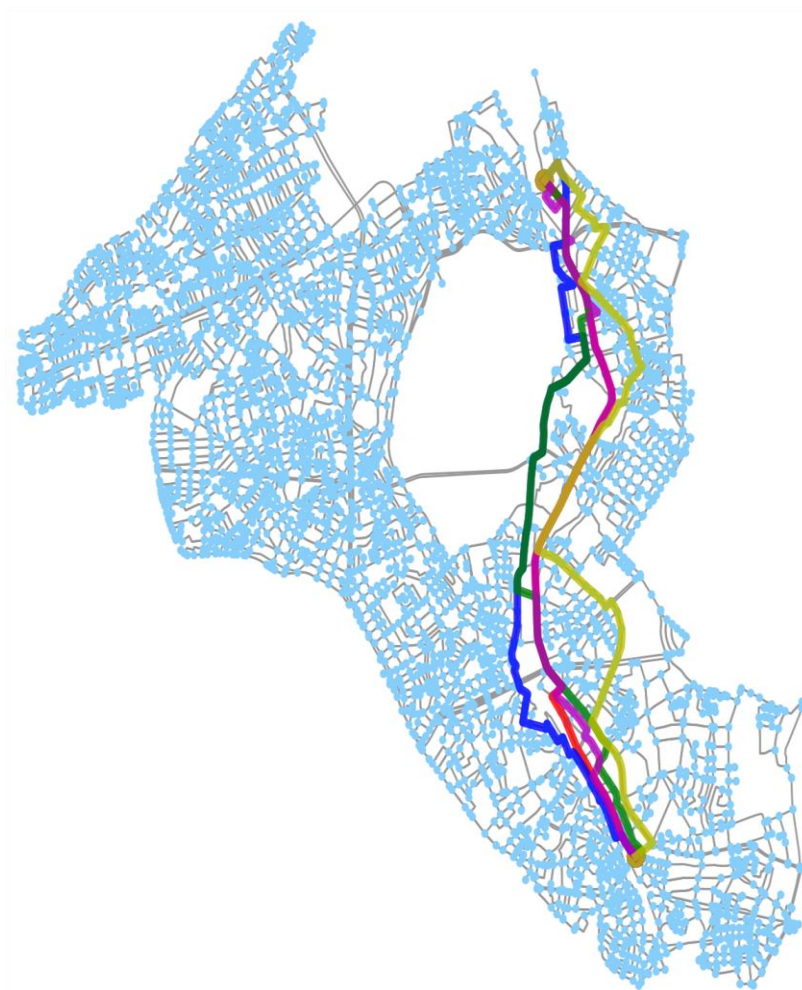
例) 代々木ー恵比寿間に5台の車を走らせる。



← 経路候補



混雑緩和後→





# 実際に体験してみる。

全体の流れ。（古典 & 量子）

1. 古典計算：地図と車のGPSデータを用意
2. 古典計算：実行に必要なデータを入力
3. **古典AI**：混雑が起きている時間帯を特定
4. 古典計算：その時間帯に走っている車に対して、代替りのルートを決定
5. 古典計算：混雑を最小化させる問題をQUBOの形に定式化
6. **古典計算 & 量子計算**  
QUBOの解を見つける（混雑が解消されるような代替りのルートを見つける）
  1. 古典計算：車のルートを再配置させる
  2. 2~6を混雑が解消するまで行う

実行環境

OS: windows, macos 言語: Python

パッケージ: xgboost, dwave-ocean-sdk, osmnx, networkx

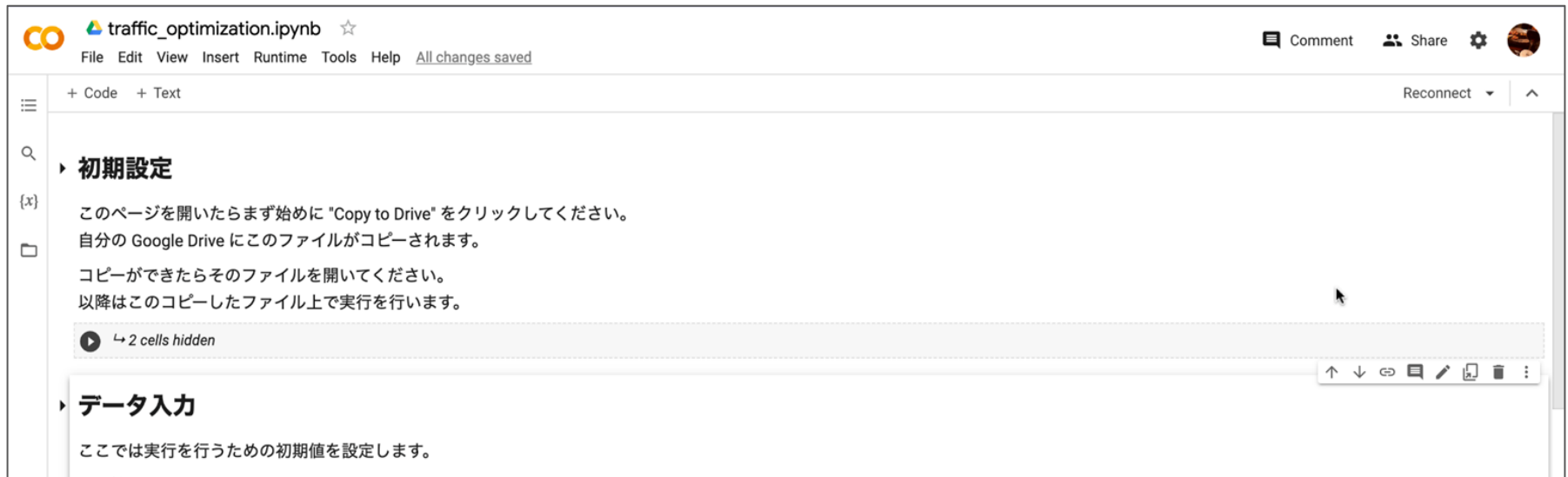
# 実際に体験してみる。

## 配布物の内容

- ・ 実際のソースコード（古典 & 量子） traffic\_optimization.ipynb

リンクはこちら↓

<https://colab.research.google.com/drive/1i8cCGeaL8Sdju1x1Aty6uCMhL20SKFRq#scrollTo=dDuYh3H9PtDT>



# 実際に体験してみる。

## Step1 古典計算：地図と車のGPSデータを用意

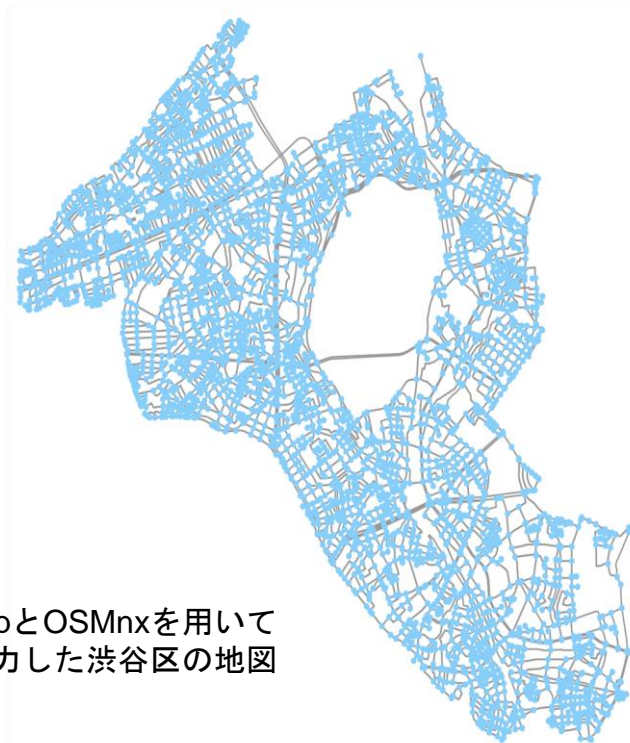
今回はオープンデータである OpenStreetmap と osmnx を用いる。

### OpenStreetMap

世界中の道路、カフェ、駅などのデータが入っている。

### OSMnx

OpenStreetMapのデータを視覚化したり分析することができる Python パッケージ。



OpenStreetMapとOSMnxを用いて  
出力した渋谷区の地図

# 実際に体験してみる。

## Step2 古典計算：実行に必要なデータを入力

ここでは今回実行を行うための条件を設定します。

デフォルトでは以下のようにになっています。

地域：渋谷区

スタート地点：代々木駅、ゴール地点：恵比寿駅

渋滞を予測する時間帯：月曜日の10時

渋滞を緩和する基準値（混雑度）：30

走らせる車の台数：4台、経路の提案数：5台



それ以外の値は量子コンピュータの細かい設定なので、今回は説明を省略します。

# 実際に体験してみる。

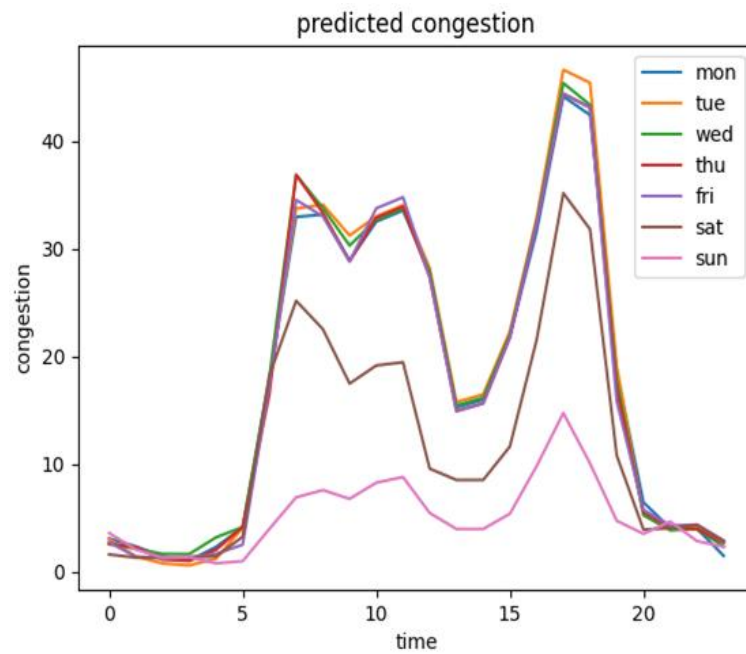
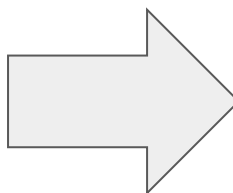
## Step3 古典AI：混雑が起きている時間帯を特定。

曜日と時間帯から各道路の混雑度を学習。

各曜日と時間帯の混雑度  
予測モデルを構築

説明変数 (入力)		目的変数 (出力)
曜日	時間帯	混雑度
月	10:00	10
金	12:00	6
火	9:00	12
金	21:00	8
土	2:00	6
...	...	...

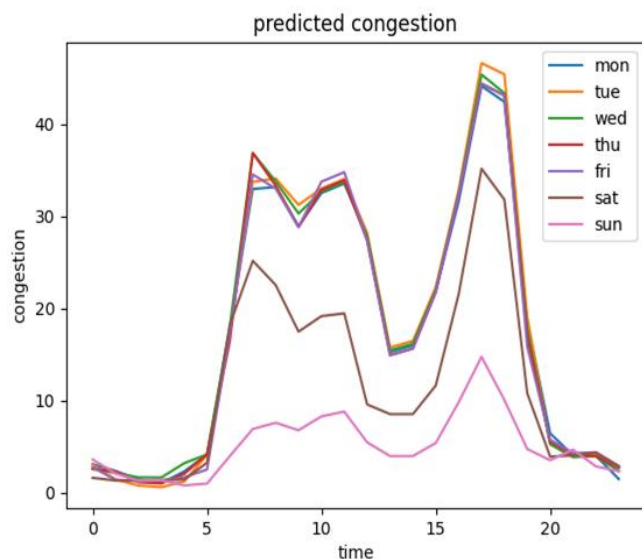
学習



# 実際に体験してみる。

Step3 古典AI：混雑が起きている時間帯を特定。

曜日と時間帯から各道路の混雑度を推定。



各曜日と時間帯の混雑度予測モデル

推定

実際のルート of 混雑度を推定

例1

曜日	時間帯
月	10:00



混雑度
10

例2

曜日	時間帯
金	12:00



混雑度
6

< 回帰することのメリット >

- ・ 統計的に誤差が小さい値を出してくれる
- ・ 訓練データにはなかった"10:30"なども推定可能

# 実際に体験してみる。

**Step3 古典計算：混雑が起きている時間帯を特定。**

今回は Step1 で取得したデータからルートを決めるために networkx を用いる。

## networkx

NetworkX は、複雑なネットワークの構造、ダイナミクス、および機能を作成、操作、および研究するための Python パッケージです。（<https://networkx.org/> より引用）

これを元に各車の目的地までのルートを計算しておく。（今回は最短距離を計算する。）

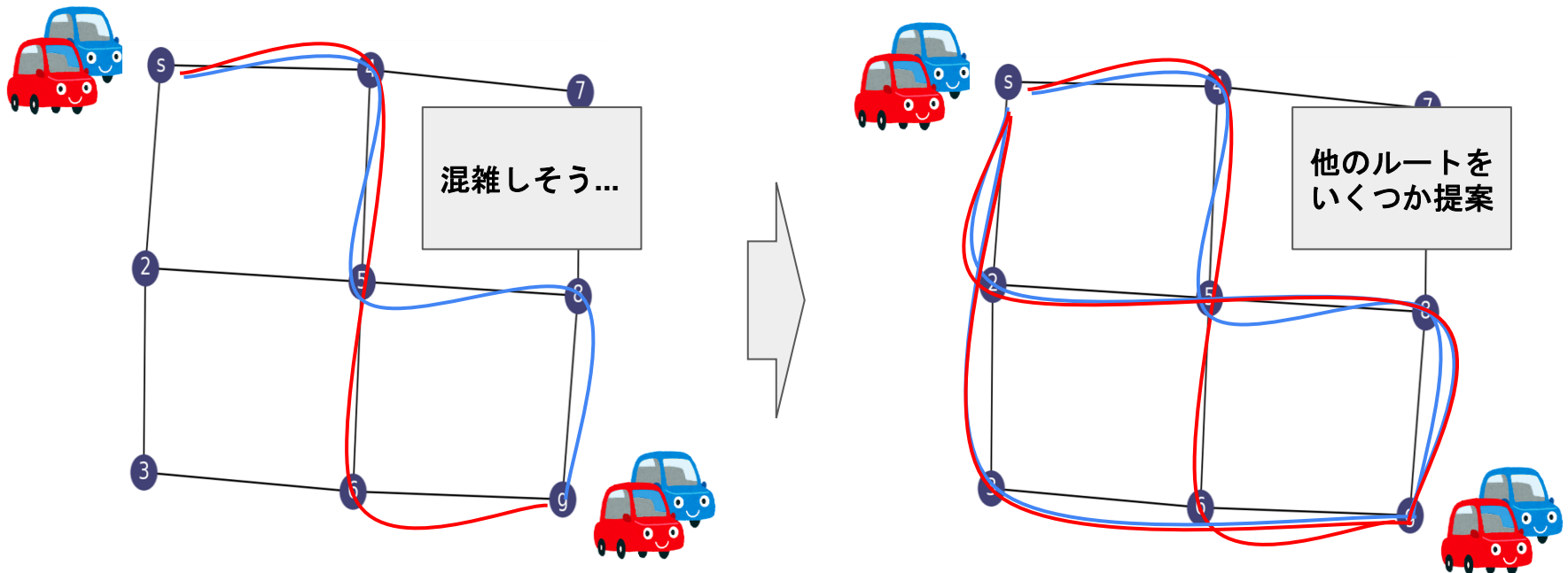
計算したルートの内、混雑が起きているルートの組を特定する。

今回は目的地までのルートを計算する部分と混雑が起きているルートの組を特定するプログラムをあらかじめ作成しているのでそれを用いる。

# 実際に体験してみる。

## Step4 古典計算：それぞれの車に対して、代わりのルートを決定

Step2で混雑が起きているルートの組に該当する車について考える。該当する車は代わりのルートをいくつか提案する。（どのルートが一番混雑しないかはわかっていない。）



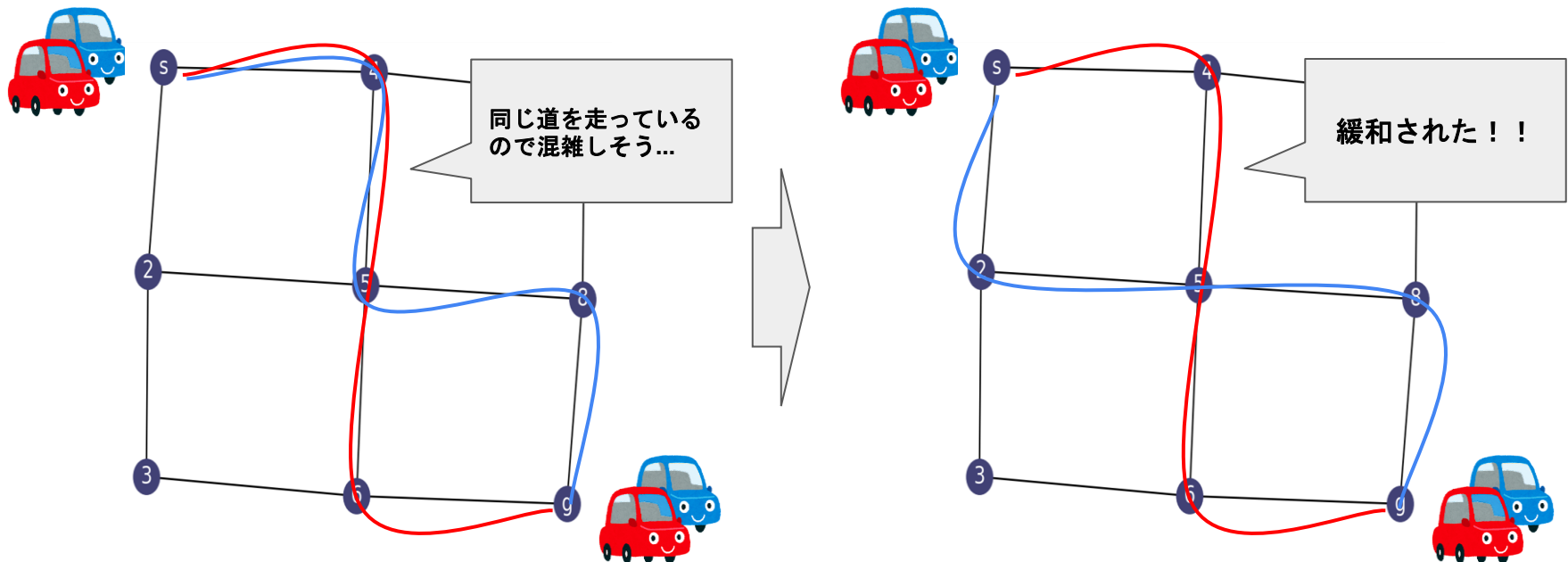


# 実際に体験してみる。

## Step5 古典計算：混雑の最小化させる問題をQUBOの形に定式化

今回は QUBO に変換するプログラムをあらかじめ作成しているのでそれを用いる。

### 混雑を最小化させる仕組み



# 実際に体験してみる。

## Step6 古典計算 & 量子計算

QUBOの解を見つける（混雑が解消されるような代わりのルートを見つける）

DWave社の量子アニーリングを用いて QUBO の解を見つける。求めた解をもとに古典コンピュータで経路を決める。

### dwave-ocean-sdk

Oceanは量子コンピュータで難しい問題を解決するための D-Wave のツールです。

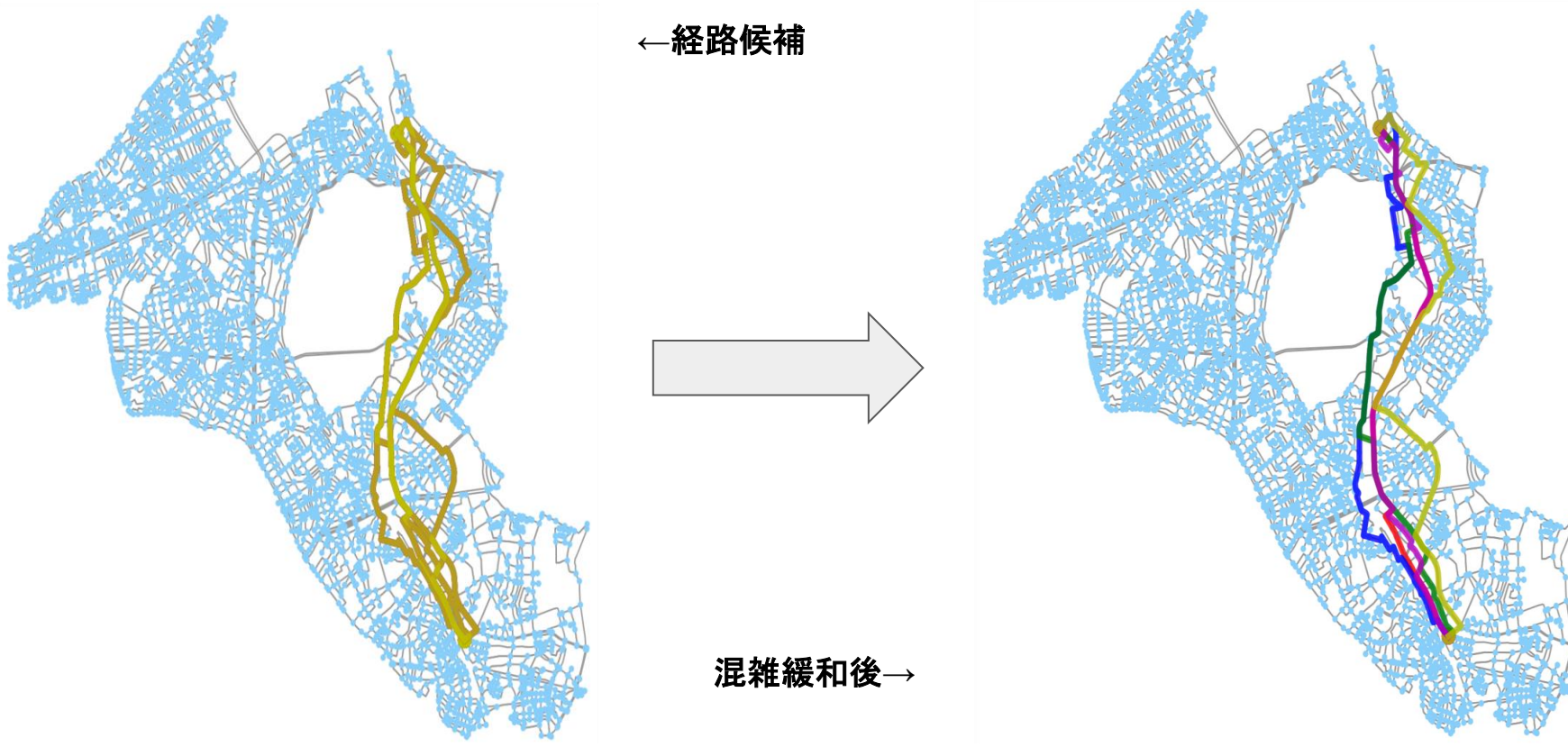
(<https://docs.ocean.dwavesys.com/en/stable/> より引用。)

これらの処理はプログラムをあらかじめ作成しているのでそれを用いる。

# 実際に体験してみる。

## Step6 古典計算 & 量子計算

QUBOの解を見つける（混雑が解消されるような代替のルートを見つける）



# 実際に体験してみる。

Step7 古典計算：車のルートを再配置させる

Step8 Step2 ~ 7 を繰り返す。

Step6 で求めた解をもとに車のルートを定める。この操作を繰り返して最も混雑がない状態を作る。

