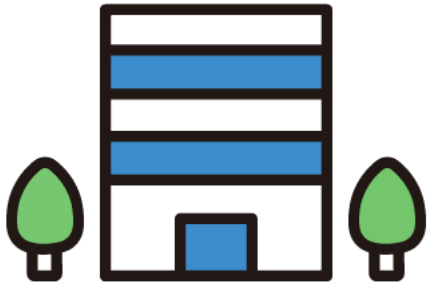
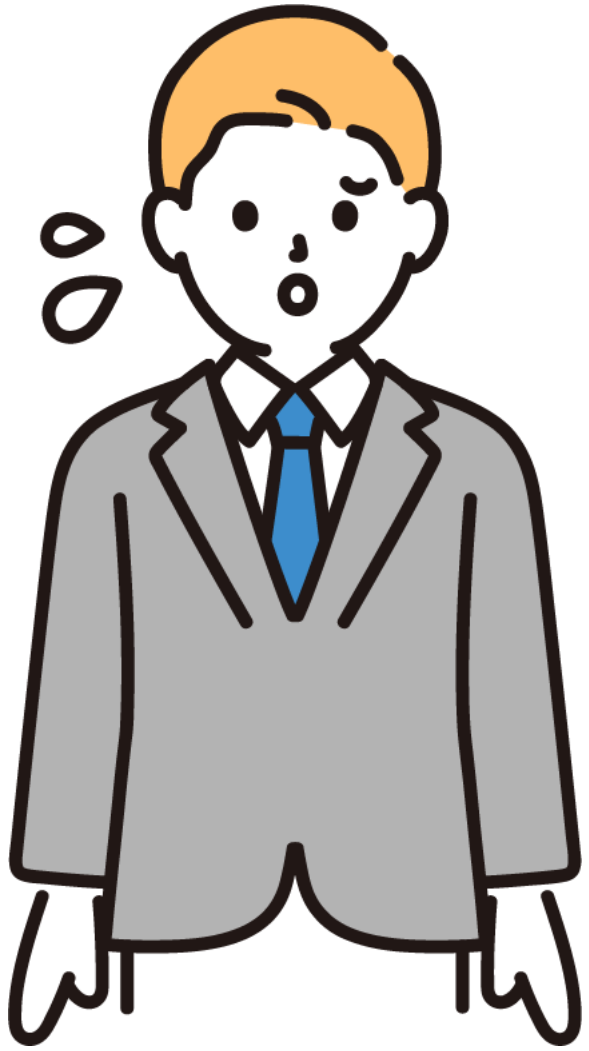
巡回セールスマン問題

Traveling Salesman Problem 最適化アルゴリズムに関する実験



８班

AJG23049　山本理
AJG23050　浅野輝心
AJG23051　高井晴
AJG23052　湯村新太
AJG23055　牧野唯希



背景

工程の効率化や**コスト削減**は産業を発展させるうえで必要不可欠

物流と配送

・商品の配送やサービスの提供において、効率的なルートを計画

旅行プランニング

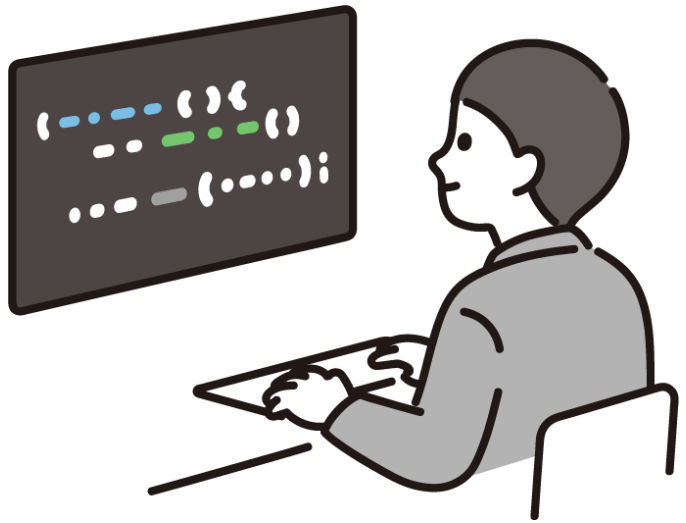
・観光資源を効率的に巡るためのルートの設計

ロボティクス

・自律移動ロボットやドローンが効果的なルートを計画

目的

- 構築法や改善法を用いて経路を**構築改善する方法**を習得する
- 計算時間から各アルゴリズムを比較する
- Pythonを用いて各手法を実装する



巡回セールスマン問題とは

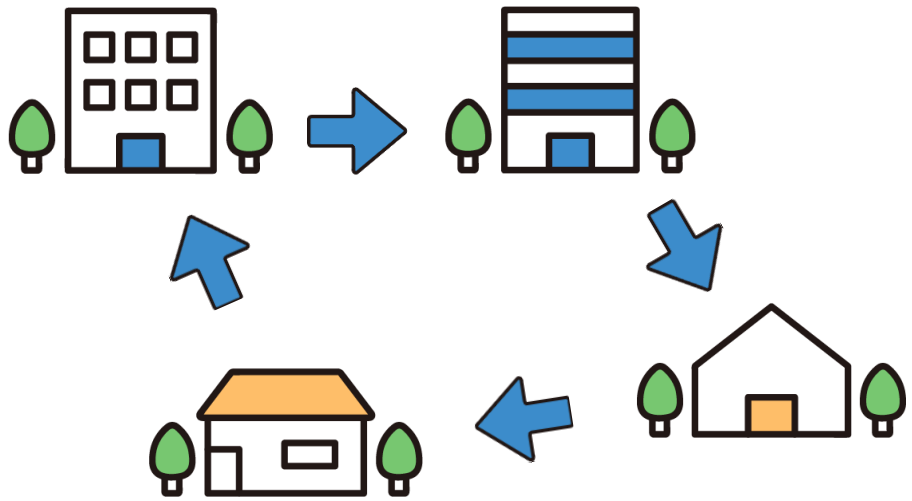
取り上げる問題

- **概要**

  ある都市から出発し、
  他のすべての都市を1度ずつ訪れ、
  出発地点に戻る

- **目的**

  総移動距離が**最小**となる
  巡回路を見つけること



巡回セールスマン問題とは

取り上げる問題

都市数が増えると計算量が膨大に

今のコンピューターでは最小値を求めることが不可能

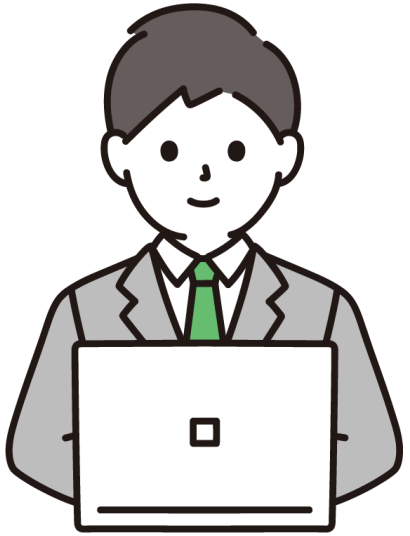
＝「NP困難」

まずは一つのルートを作りそれを最適化をしていく

実験方法

０．都市の生成
１．最近隣法
２．2-opt法
３．多出発局所探索
４．大域的最適解の探索法
　　分枝限定法

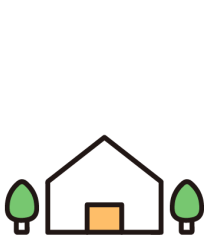
計算時間
総移動距離の改善

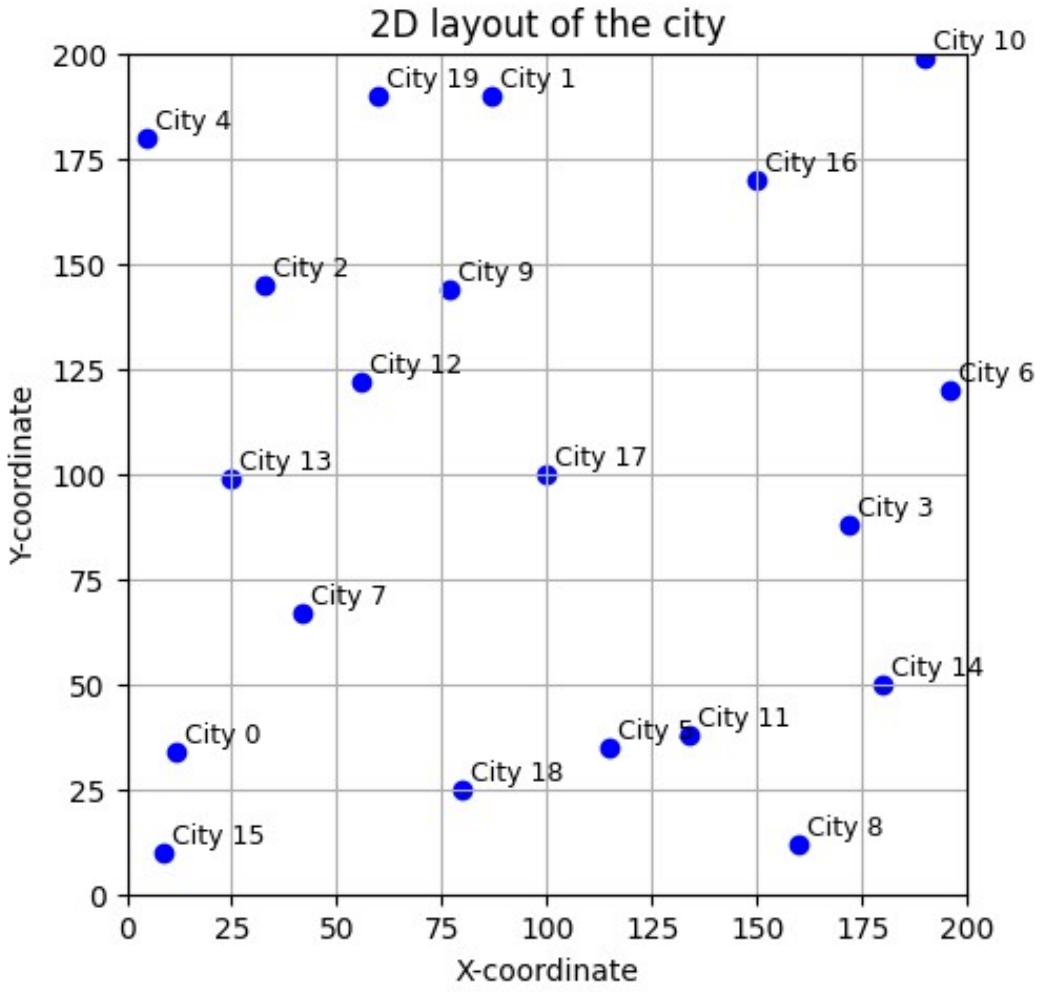


0．都市の生成

〇手順

- ランダムに20の都市を生成
- 重なりが無いように注意
- 200×200の平面座標

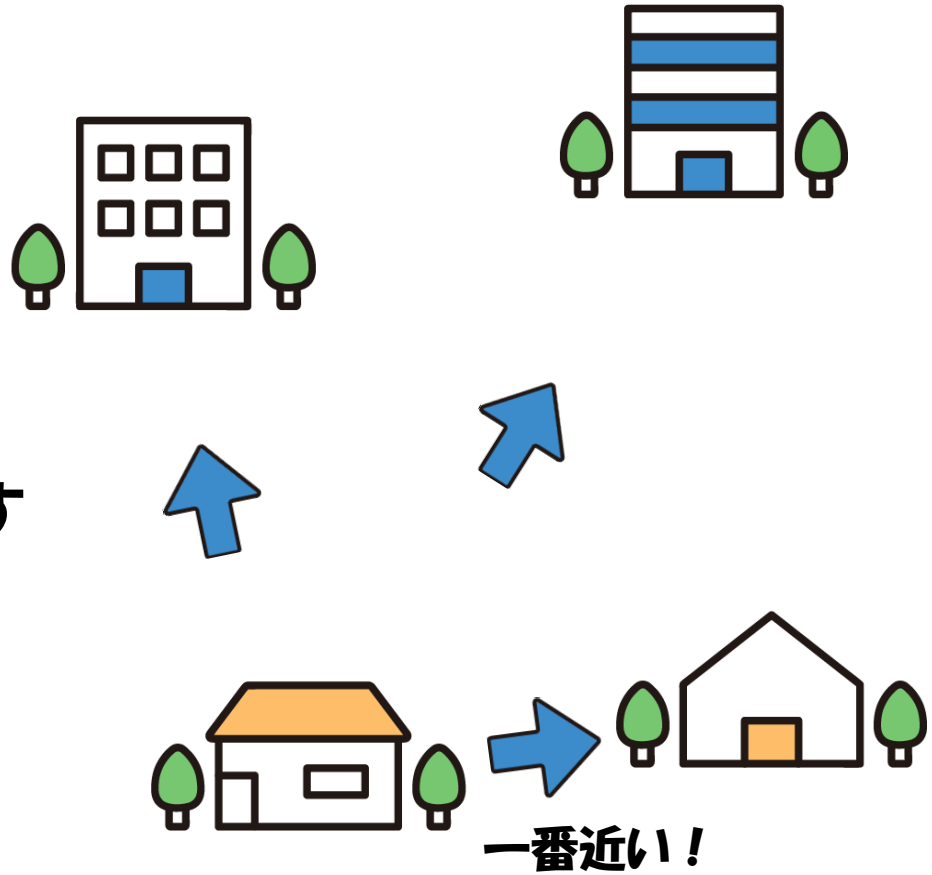




１．最近隣法

〇手順

・出発都市を1つ選ぶ
・現在地から最も近い未訪問の都市へ移動
・これをすべての都市を訪れるまで繰り返す
・最後に出発点に戻って巡回路を完成



２．2-opt法

〇概要

・既存の巡回路から、任意の2本の枝（経路）を選んで付け替えることで、新たな巡回路を構成

〇手順

・最近隣法で初期解を求める

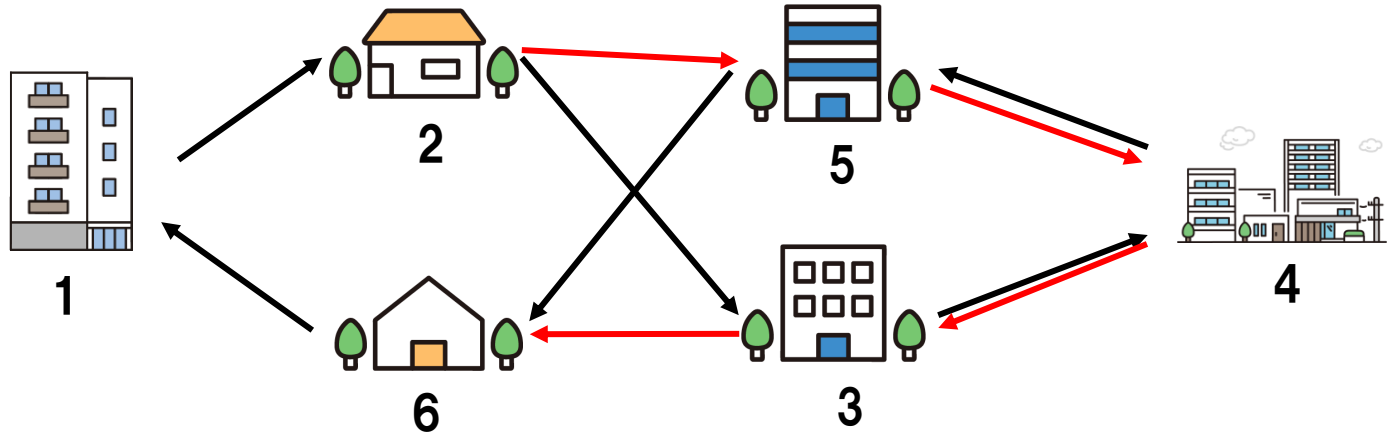
・2本の枝を入れ替えた場合の全候補を評価

・移動距離が最も短くなる組み合わせを選んで巡回路を更新

・この操作を改善ができなくなるまで繰り返す

イメージ図

2-opt法



◆元の経路

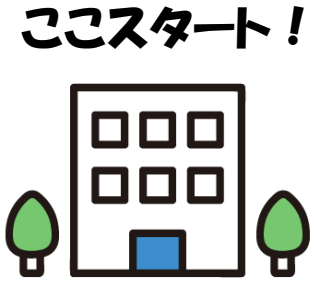
1 ー 2 ー 3 ー 4 ー 5 ー 6 ー 1

◆変更後の経路

2-3, 5-6の経路を付け替え

1 ー 2 ー 5 ー 4 ー 3 ー 6 ー 1

３．多出発局所探索



〇概要
- １や2の方法は一つの局所解にとどまる可能性がある
- 初期解を変えて複数回探索を行い、より良い解を探索
- 最良の巡回路をその中から選ぶ

〇手順
- 初期解の生成を以下の２通りで行う
  １．各都市を出発点とした最近隣法
  ２．ランダムな巡回順序
- 各初期解に対して2-opt法を適用

４．Held-Karp法

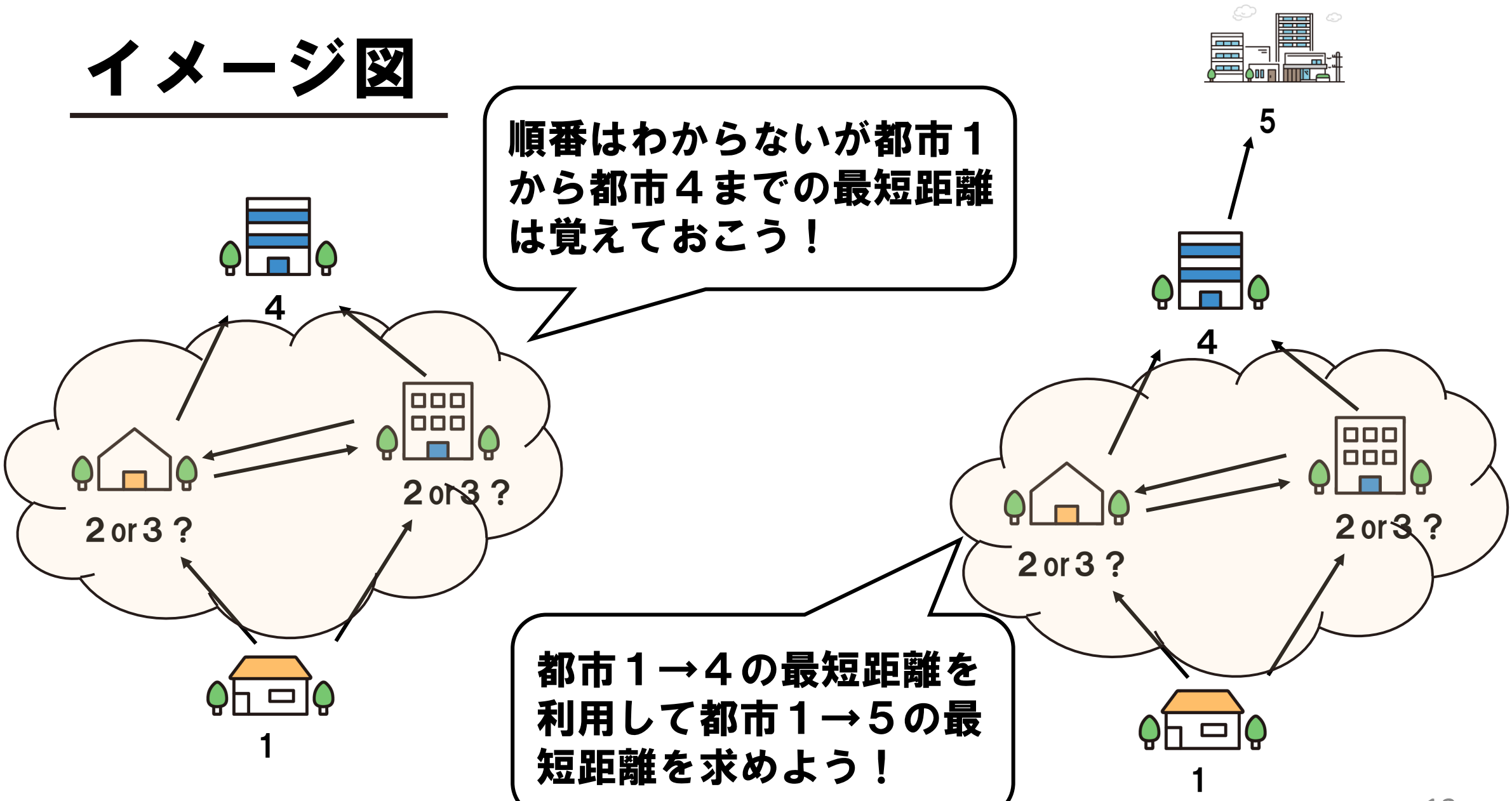
〇概要

基本的に行うことは全探索であるが一度訪れたことのあるルートを残しておくことで重複をなくした探索方法

〇手順

1. **訪問済みの都市を探す。**
2. **探した訪問済みの都市からすべての未訪問の都市への距離を探索し保存する。**
3. **手順１,２を探索が終わるまで実行**

イメージ図


順番はわからないが都市１から都市４までの最短距離は覚えておこう！
4
2 or 3 ？
2 or 3 ？
1
都市１→４の最短距離を利用して都市１→５の最短距離を求めよう！
5
4
2 or 3 ？
2 or 3 ？
1

効率

・訪問済みの都市をビット列の0,1で表現する。都市が５つの場合次のようになる。
（都市２と５が訪問済みの場合）

都市５	都市４	都市３	都市２	都市１
1	0	0	1	0
訪問済み			訪問済み	

N個の都市の訪問状態のすべての総数はN桁のビット列で表現されるため総数は2^N通り
それぞれにN^2回の演算を行うので演算回数は　　$2^N \times N^2$

総当たりの演算回数は　　N!　　なので都市数が２０個の時、効率は

$$\frac{2^{20} \times 20^2}{20!} = 1.723 \times 10^{-10} = 1.723 \times 10^{-8}[\%]$$

分枝限定法

〇概要

・全探索を効率よく省略しながら最適解を見つける。

〇手順

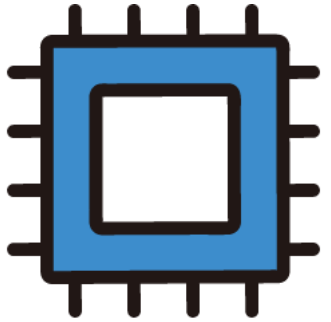
・探索空間を「部分問題」に分割

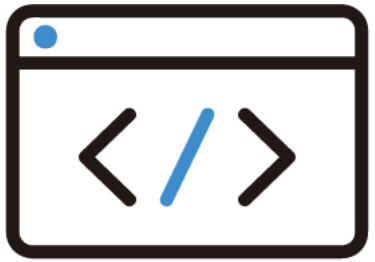
・各部分問題ごとに下界を計算

・その下界が現在の最良解よりも悪い場合は探索を打ち切る。

実験方法

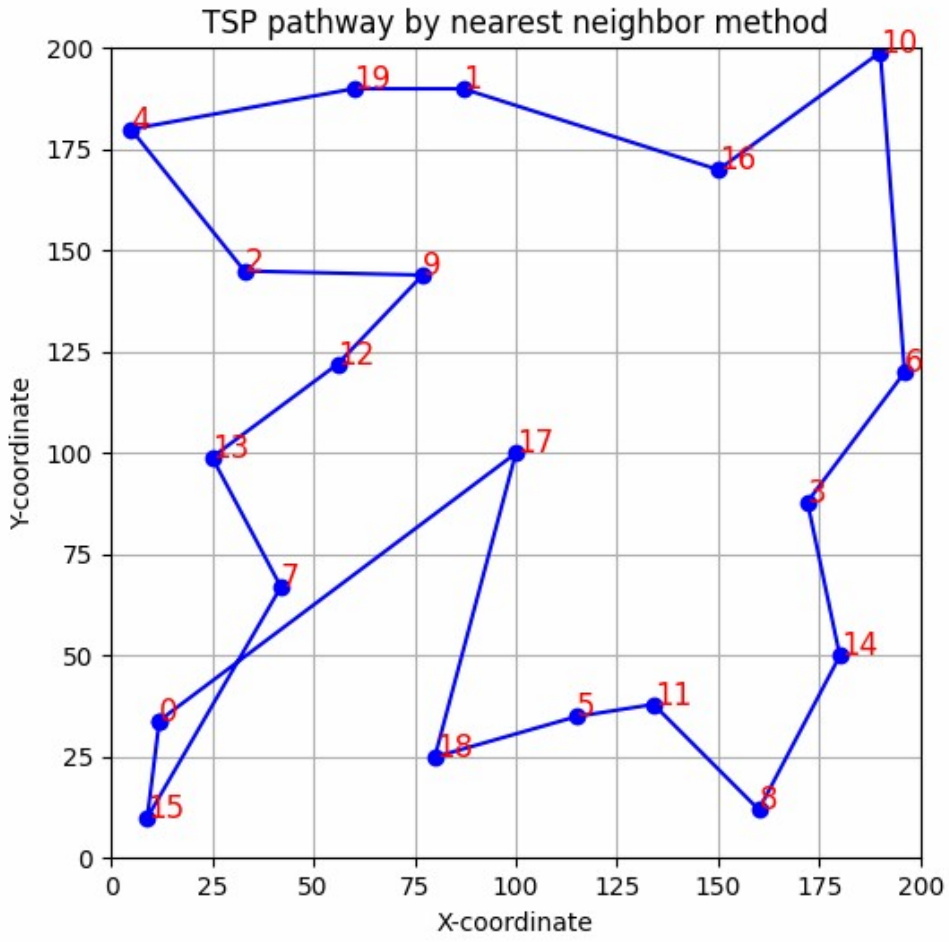
- CPU：AMD Ryzen 7 5825U
- RAM：16.0 GB（15.4 GB 使用可能）
- 使用エディター：Visual Studio Code





結果【最近隣法】

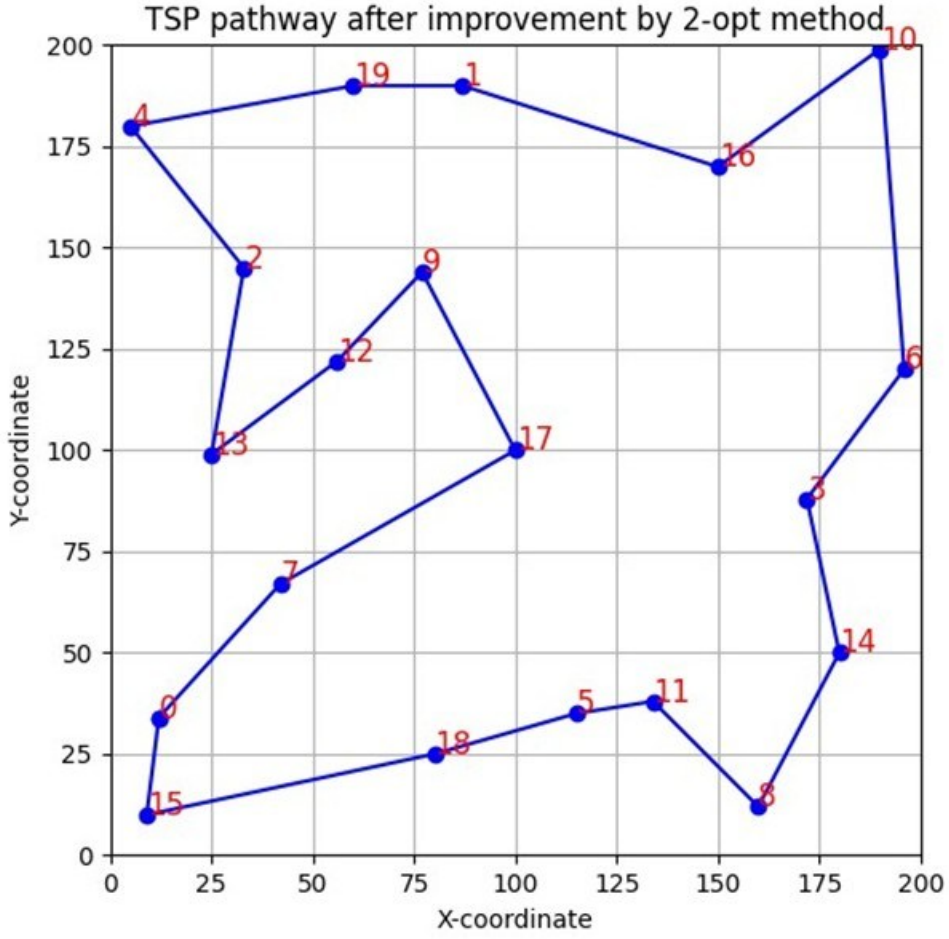
総移動距離：963.57
実行時間：0.000018秒



※画像は一例（分かりやすさのため、都市の位置を固定）

結果【2-opt法】

総移動距離：910.07
実行時間：0.007701秒
2-opt法の総反復回数：7
距離改善：53.50(5.55%減少)

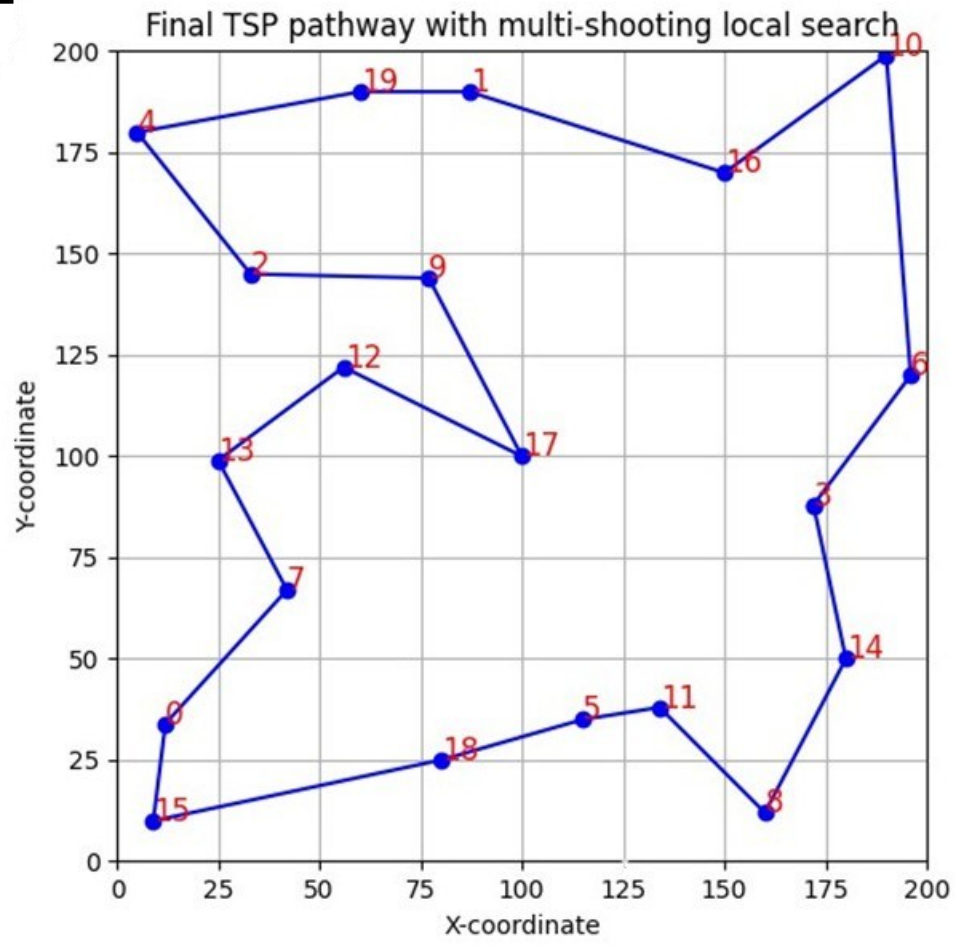


※画像は一例（分かりやすさのため、都市の位置を固定）

結果【多出発局所探索】

総移動距離: 895.68
実行時間：0.257616秒
通常の2-optからさらに改善: 14.39（1.58%）
最近隣法からの総改善: 67.89（7.05%）

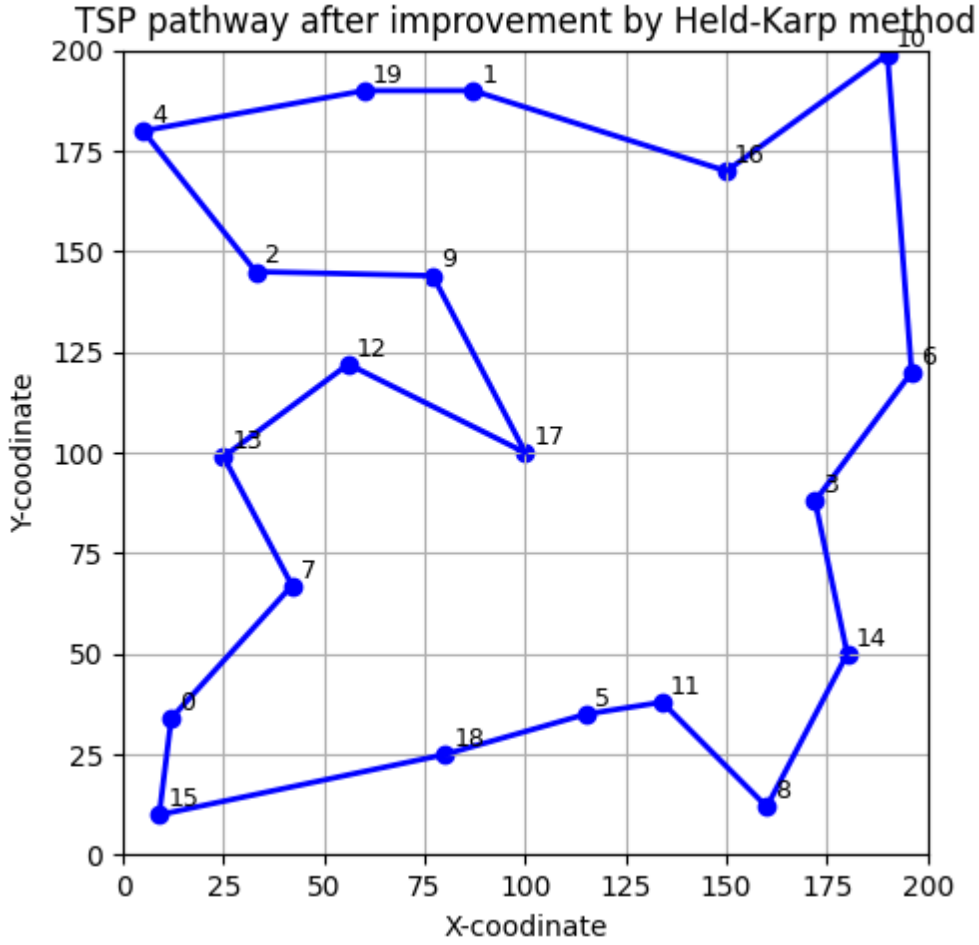
ただし、場合によっては
異なる出力結果が生じた



※画像は一例（分かりやすさのため、都市の位置を固定）

結果【 Held-Karp法】

総移動距離: 895.68
実行時間： 62.2539秒



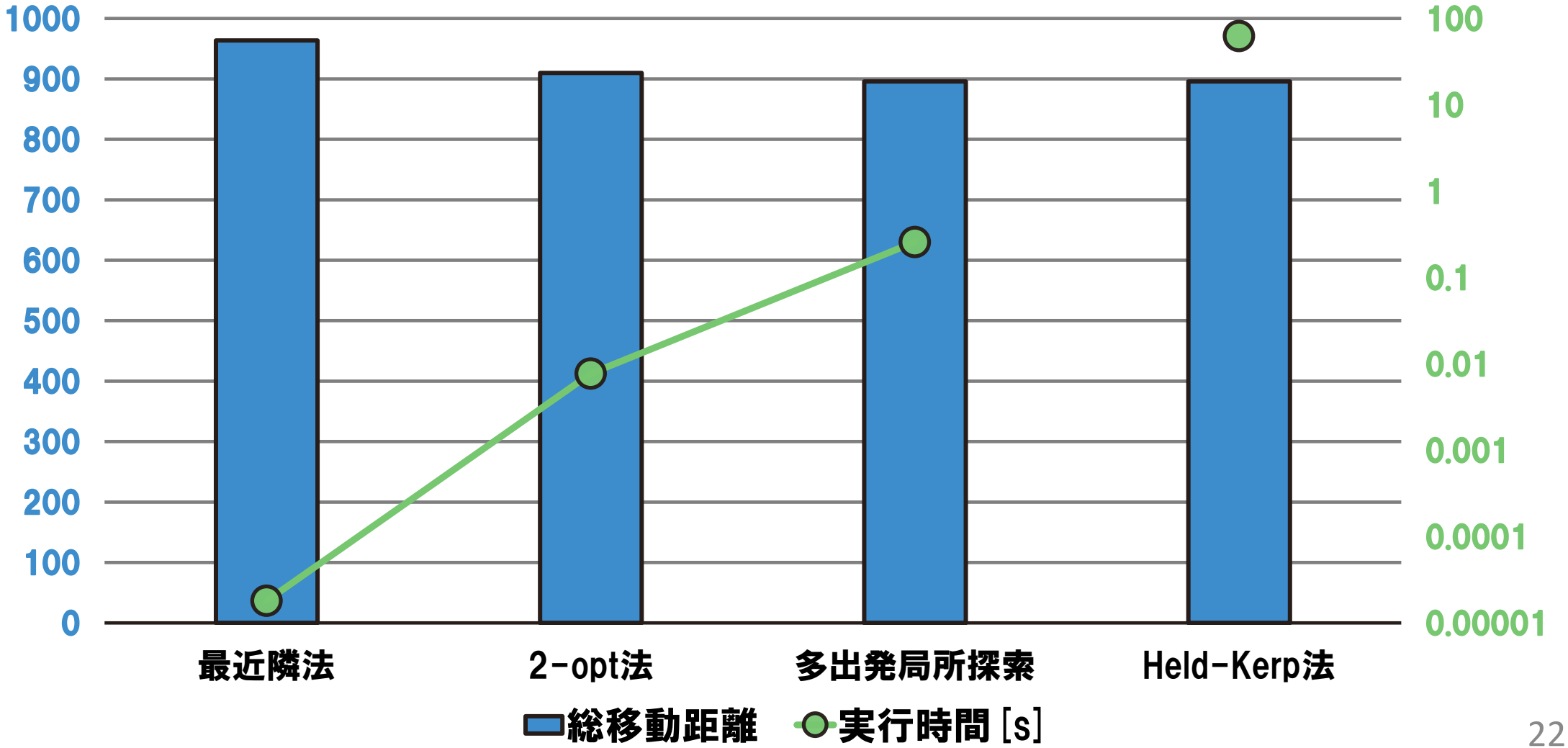
※画像は一例（分かりやすさのため、都市の位置を固定）

結果【分枝限定法】

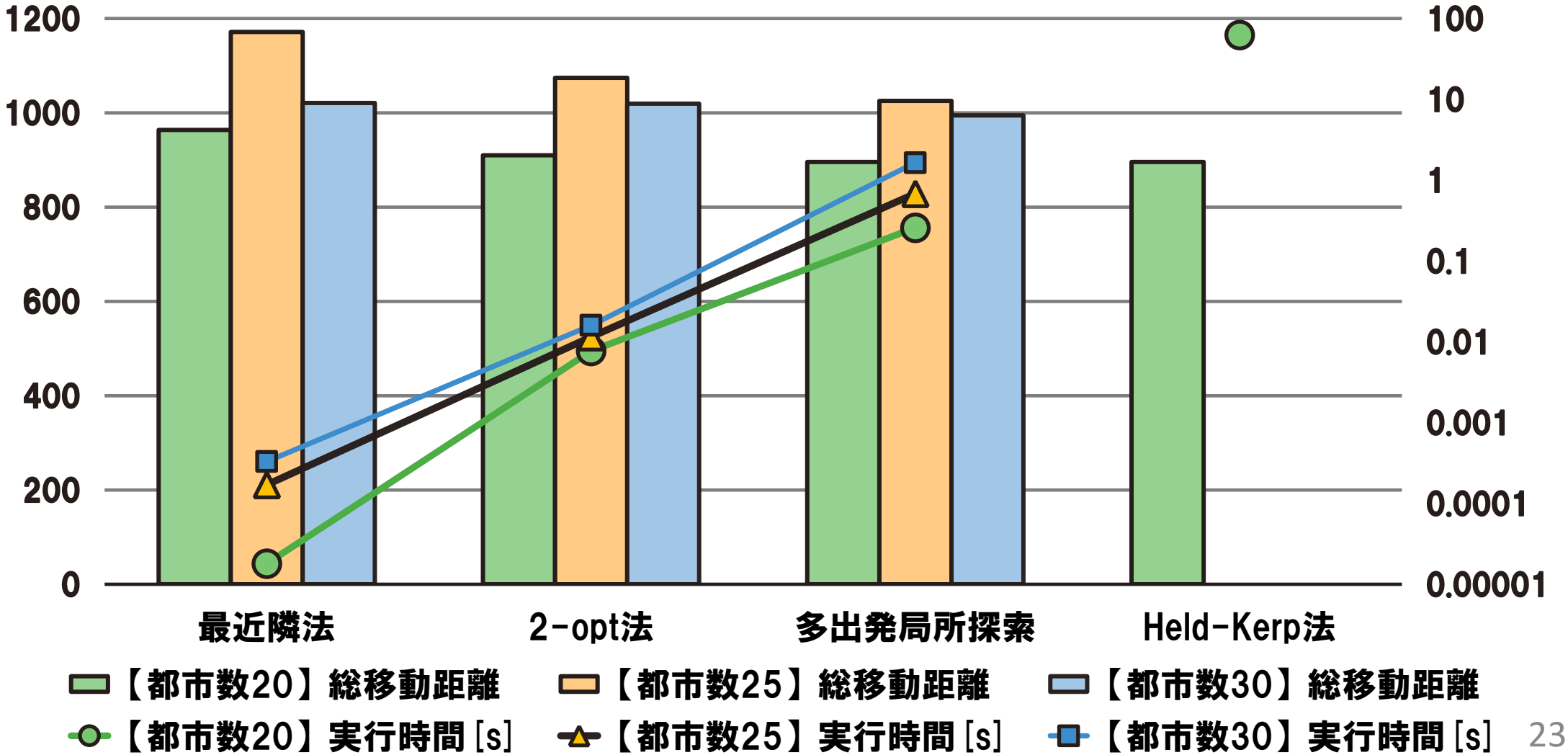
総移動距離：測定不能
実行時間：測定不能

原因：処理時間が長すぎたため。
O(n!)になる。

結果【まとめ】


1000
900
800
700
600
500
400
300
200
100
0
100
10
1
0.1
0.01
0.001
0.0001
0.00001
最近隣法
2-opt法
多出発局所探索
Held-Kerp法
総移動距離
実行時間 [s]

結果【都市数25&30】


1200
1000
800
600
400
200
0
100
10
1
0.1
0.01
0.001
0.0001
0.00001
最近隣法
2-opt法
多出発局所探索
Held-Kerp法
【都市数20】総移動距離
【都市数25】総移動距離
【都市数30】総移動距離
【都市数20】実行時間 [s]
【都市数25】実行時間 [s]
【都市数30】実行時間 [s]

考察

- 最近隣法、2-opt法、多出発局所探索と最適化を進めていくたびに結果の改善が見られた。

- 都市数20の場合、Held-Karp法と多出発局所探索では総移動距離において同じ結果が得られたので、これが最適解と考えられる。

- 実行時間は指数関数的に増加し、都市数が増えても同じ傾向が見られた。

- 分岐限定法では処理が長すぎたため、結果を得ることはできなかった。

まとめ

・工程の効率化やコスト削減を行うときに考える、巡回セールスマン問題について、取り組んだ。

・この問題はNP困難のため、あるルートを最適化させていくことで結果を導いた。

・最適化に伴い、総移動距離の短縮に成功したが、処理時間は指数関数的に増加した。

・都市数が増加すると、最適解の探索法では探索できなくなった。