# 巡回セールスマン問題に対する発見的解法の高速化

4495017　岩倉　行信（沼田研究室）

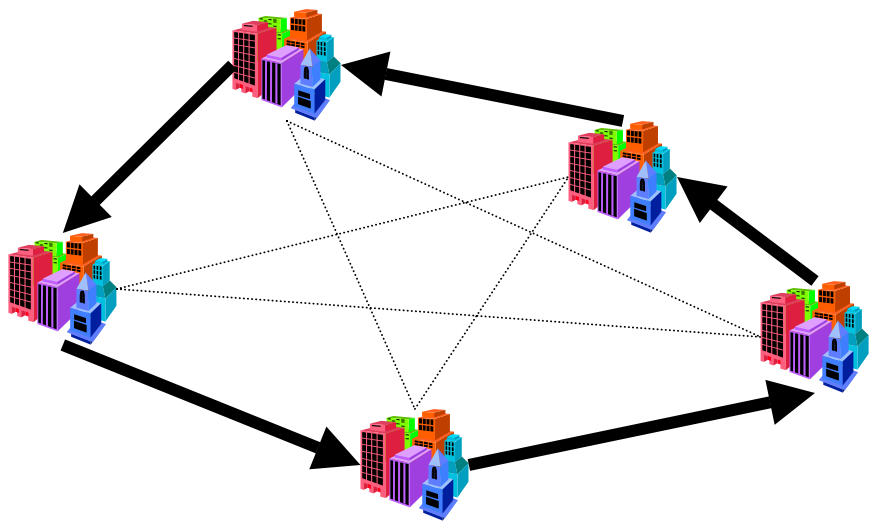
発表構成

1. はじめに
2. Candidate Setの構成法
3. 2－opt法
4. 実験結果
5. おわりに

# 1．はじめに

巡回セールスマン問題（Traveling Salesman Problem：TSP）

与えられたいくつかの都市を全て一度ずつ訪問し最初の都市に戻ってくる巡回路の中で、移動時間や距離に基づく総費用が最小となるものを求める問題



：都市

：巡回路

TSP5都市の例

# TSPの解法

発見的解法

最適解に近い解を求める　→　準最適解

（遺伝的アルゴリズム，２－opt法　etc）

多数回の探索を行い、より最適解に近い準最適解を得る。

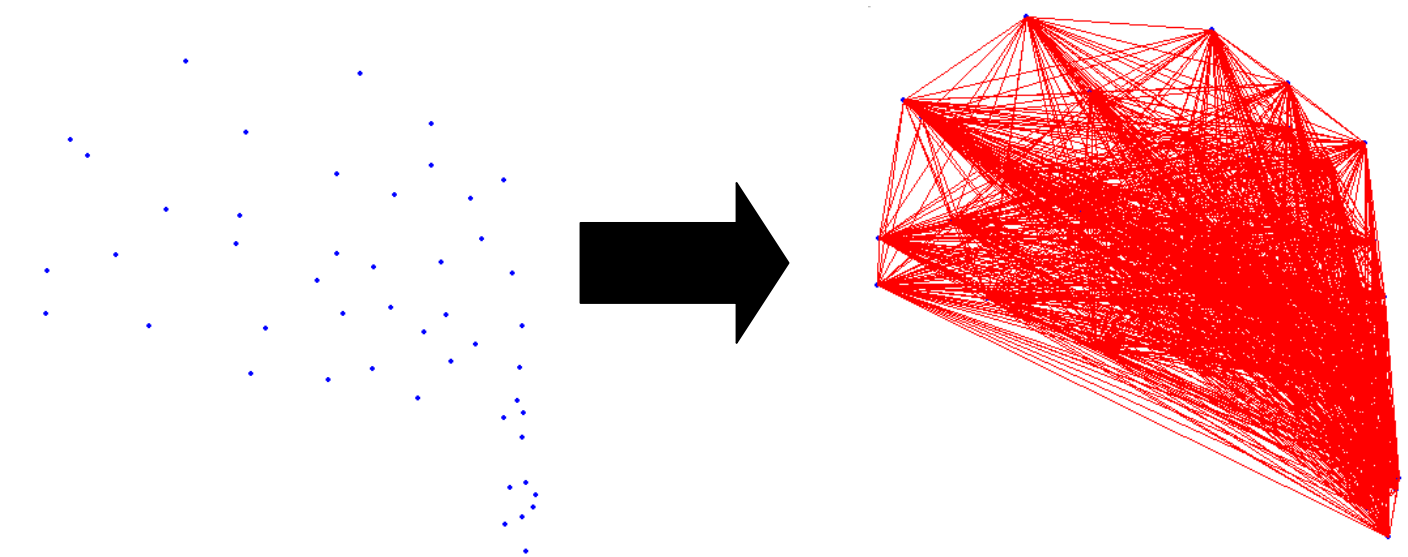
１回の探索をなるべく速く行いたい

発見的解法高速化の必要性

# 高速化の方法

通常の発見的解法

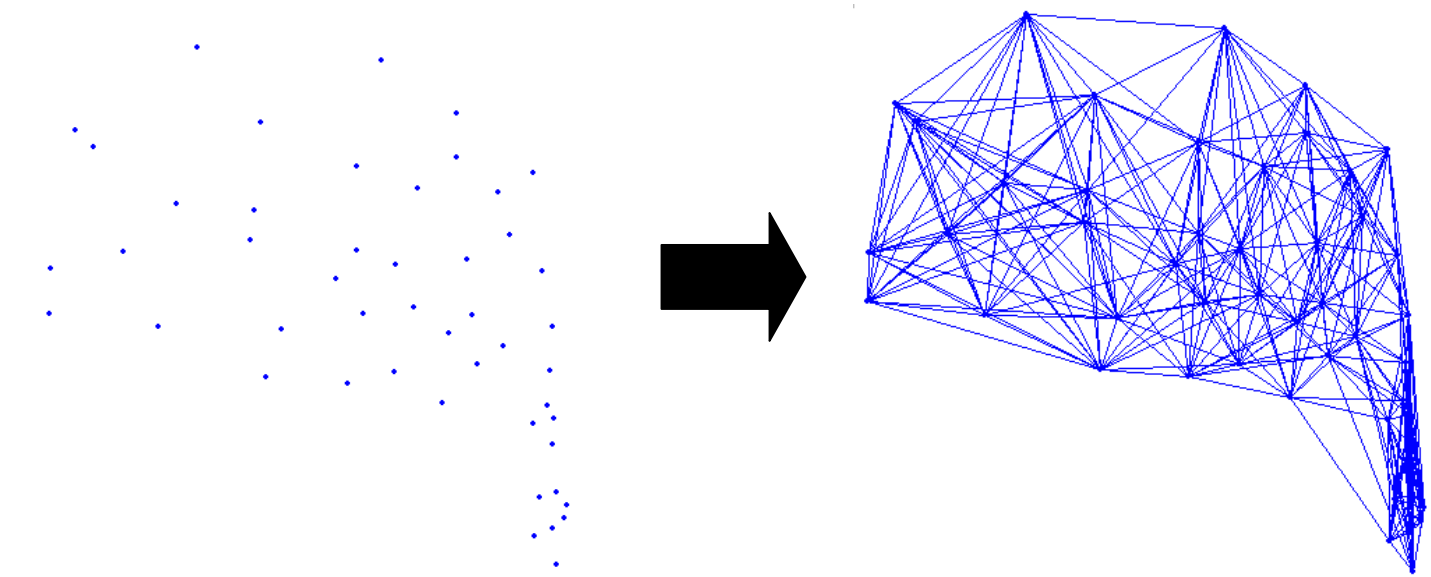
→ 巡回路の枝候補を，全都市間を結ぶ全ての枝とする



明らかに最適解に含まれないと考えられる長い枝についてまで考慮しているので，効率が悪く時間がかかる

最適解に含まれる可能性の高い枝に注目

枝を選び出し，その枝のみを巡回路の枝候補とする



考慮する枝数が減少しているので，発見的解法に必要な計算時間が短くなる

発見的解法の高速化

## Candidate Set

最適解に含まれる可能性の高い枝を選び出すことを考え、その選ばれた枝の集合

## 研究目的

発見的解法高速化の手段として，Candidate Setを取り上げる．発見的解法として２－ｏｐｔ法を用い，全枝からなる問題とCandidate Setの枝からなる問題を解き，得られる準最適解の精度と実行時間を比較する。

# ２．Candidate Setの構成法

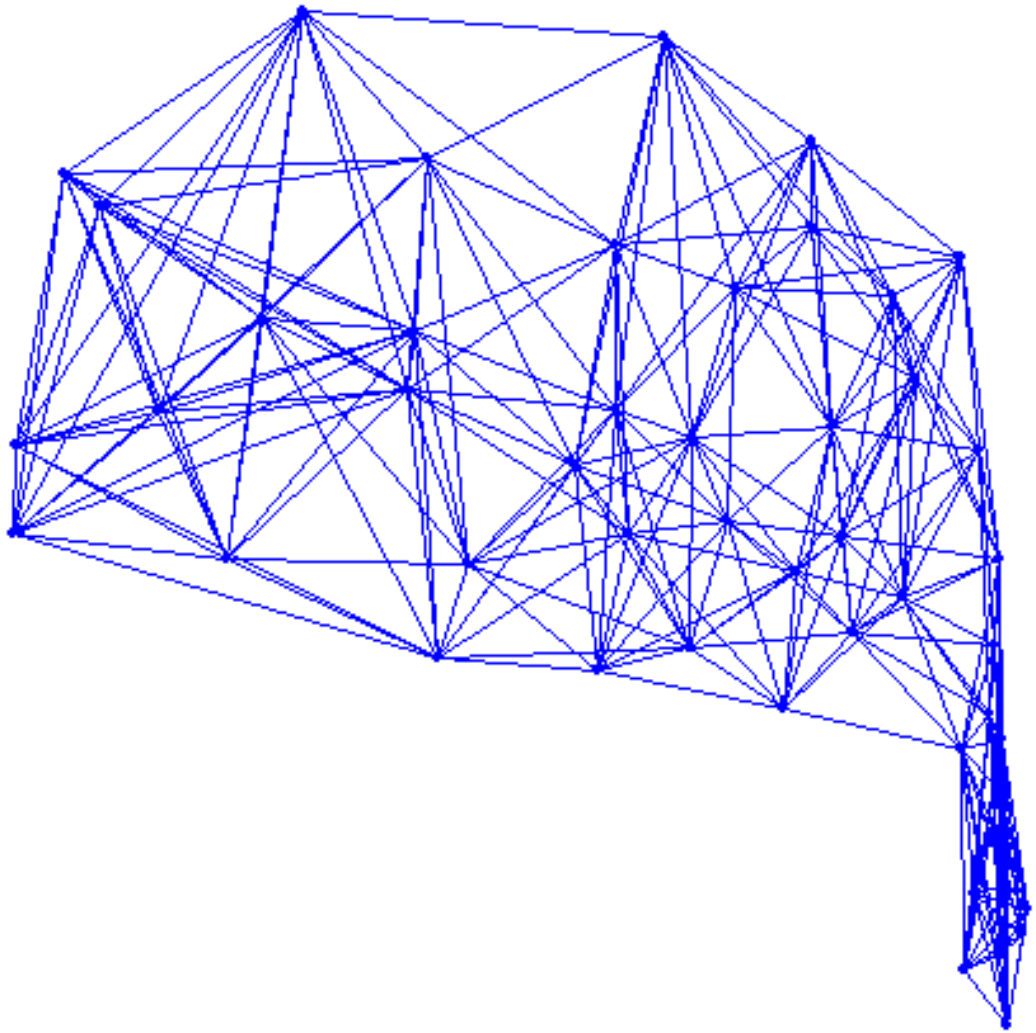
## ２．１ 方法[１] k nearest neighbors

最適巡回路では、最も近い２都市を結ぶ枝が多く含まれている。また，他の枝も近い都市間を結んでいる場合が多い

各都市についてその都市となるべく近い都市を結ぶ枝をCandidate Setとして選ぶ。

**k nearest neighbors**

ある都市に対して近い方からｋ（ｋ＞１）個までの都市



Candidate Set（方法[1]（k＝10））

k nearest neighborsは全ての都市間の移動費用を列挙することで求められる

都市数が多くなると，時間がかかる

移動費用＝ユークリッド距離の問題

ドロネーグラフの接続関係を利用することで，高速にk nearest neighborsを求めることができる

## ２．２ ボロノイ図とドロネーグラフ

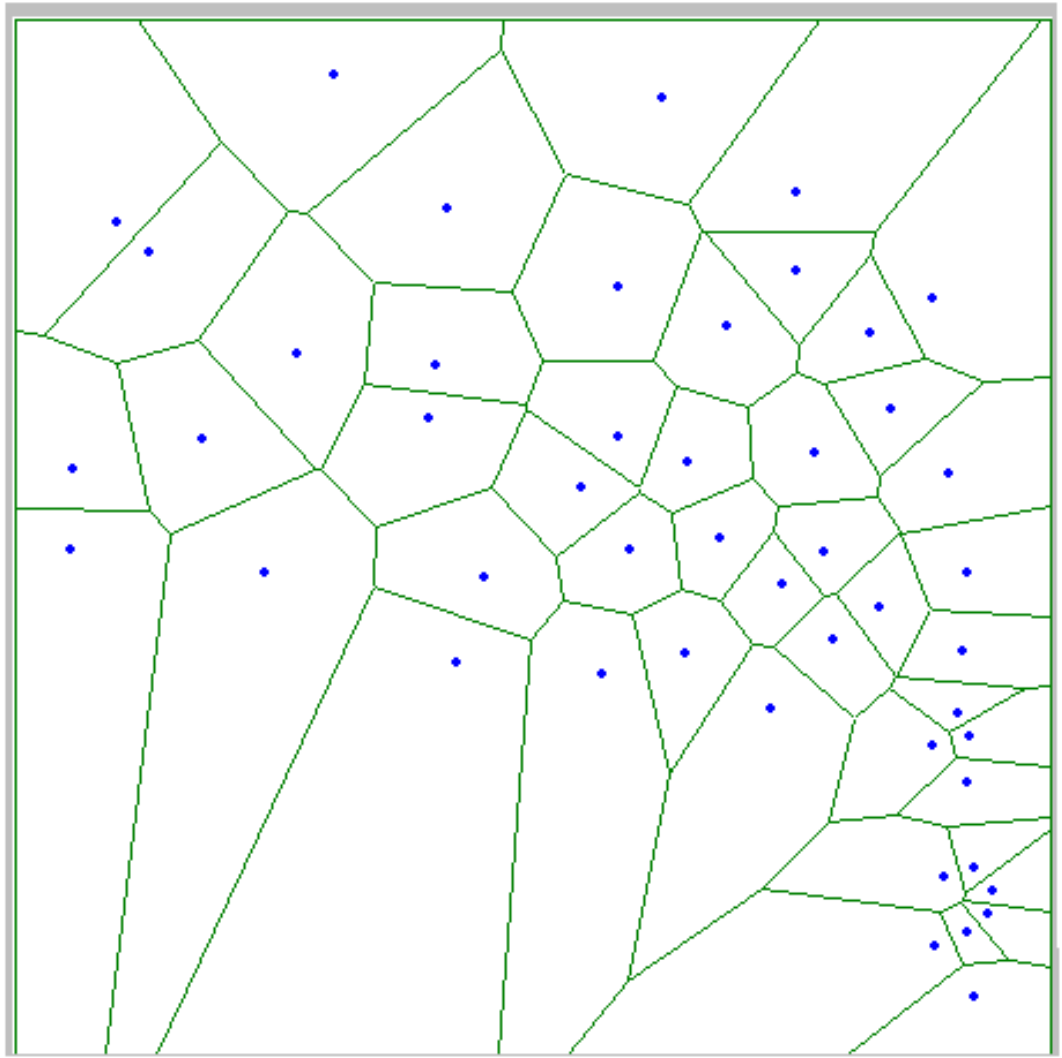
### ボロノイ図

平面上にｎ個の点Ｐｉ＝（ｘｉ，ｙｉ）　（ｉ＝１,･･･,ｎ）が与えられたとき、以下の式で与えられる点の集合を母点Ｐｉのボロノイ領域（勢力圏）と呼ぶ。

$$V(P_i) = \prod_{i \neq j} \{P = (x, y) \mid d(P_i, P) \leq d(P_j, P)\}$$

（ただし，$d(P_i, P)$は点$P_i$と点$P$のユークリッド距離）

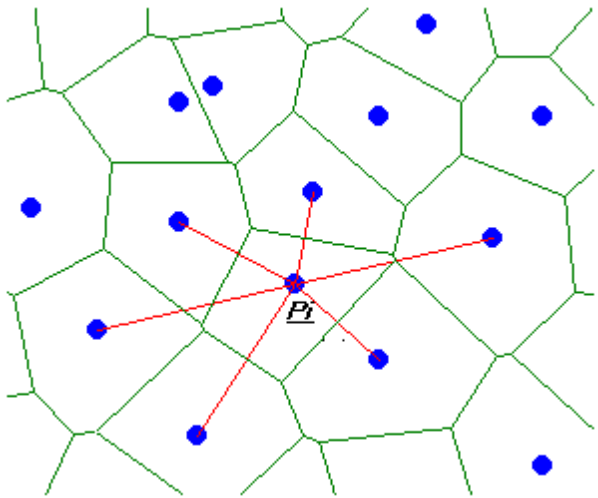
Ｐ1，････，Ｐｎの中でＰｉが最近点である集合



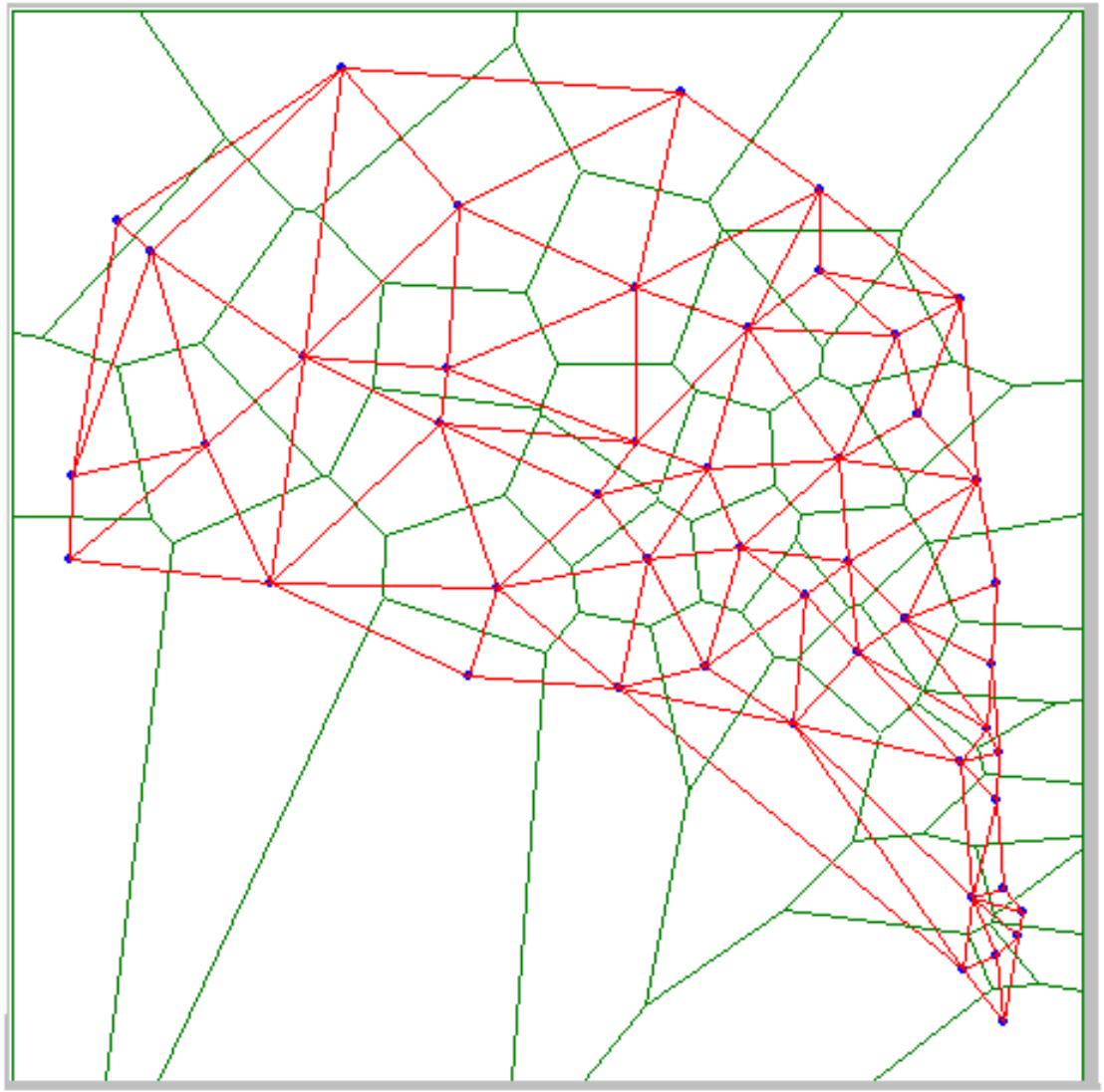
ボロノイ図（４８都市問題）

## ドロネーグラフ

ボロノイ図において母点$P_i$と母点$P_j$のボロノイ領域が同じ辺を共有しているとき、母点$P_i$と母点$P_j$のボロノイ領域は隣接関係にあるという。
ボロノイ領域（母点）を点とし、隣接するボロノイ領域間を枝で結んだグラフをドロネーグラフと呼ぶ

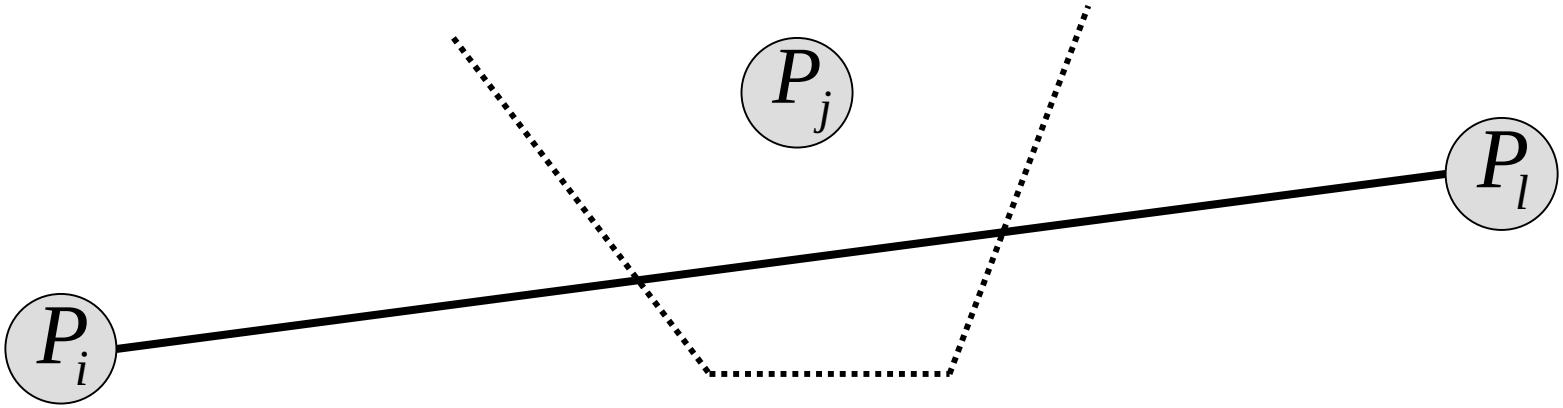



隣接するボロノイ領域



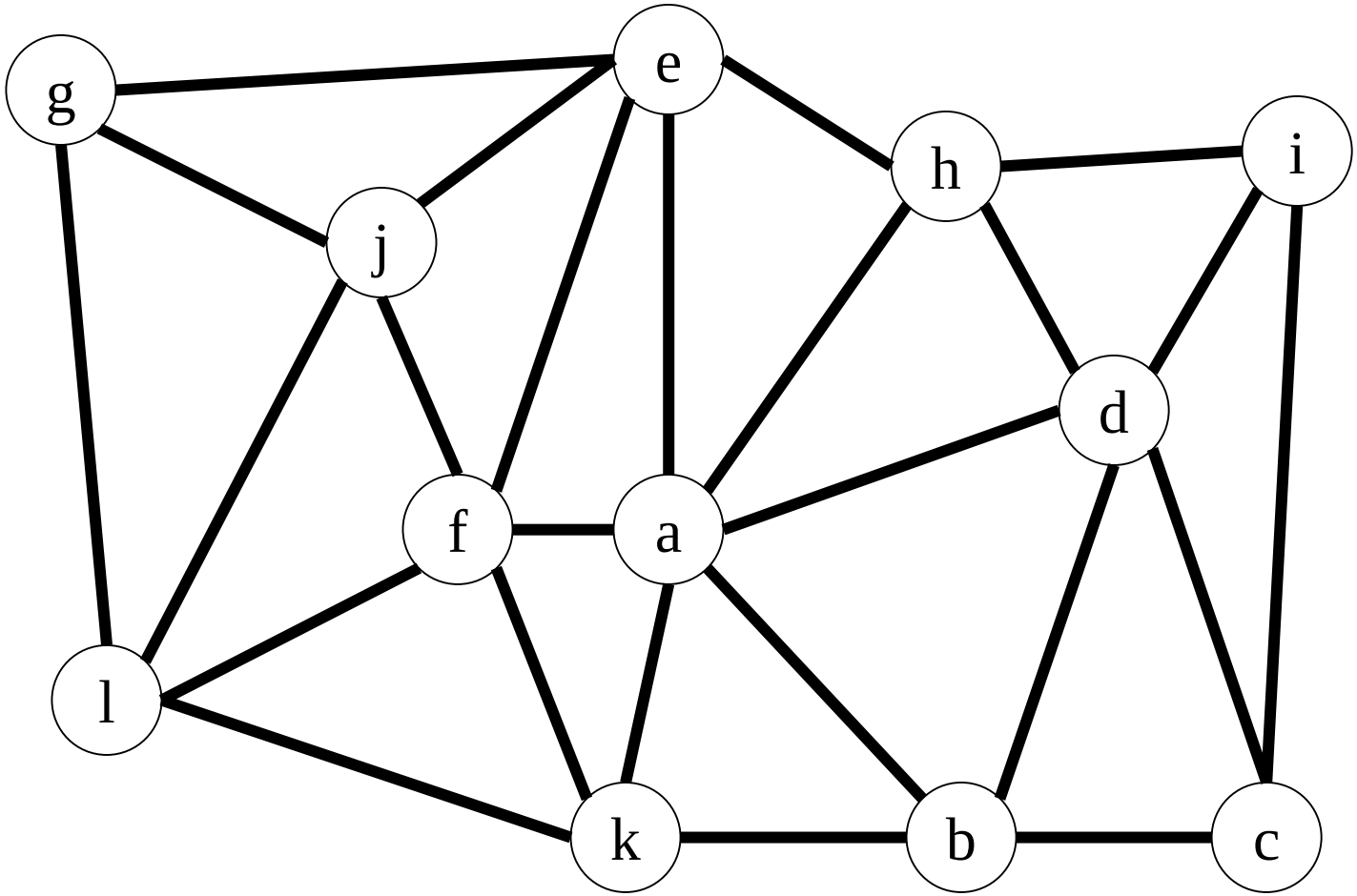
ドロネーグラフ（４８都市問題）

# ２．３ドロネーグラフの接続関係を利用してのk nearest neighbors

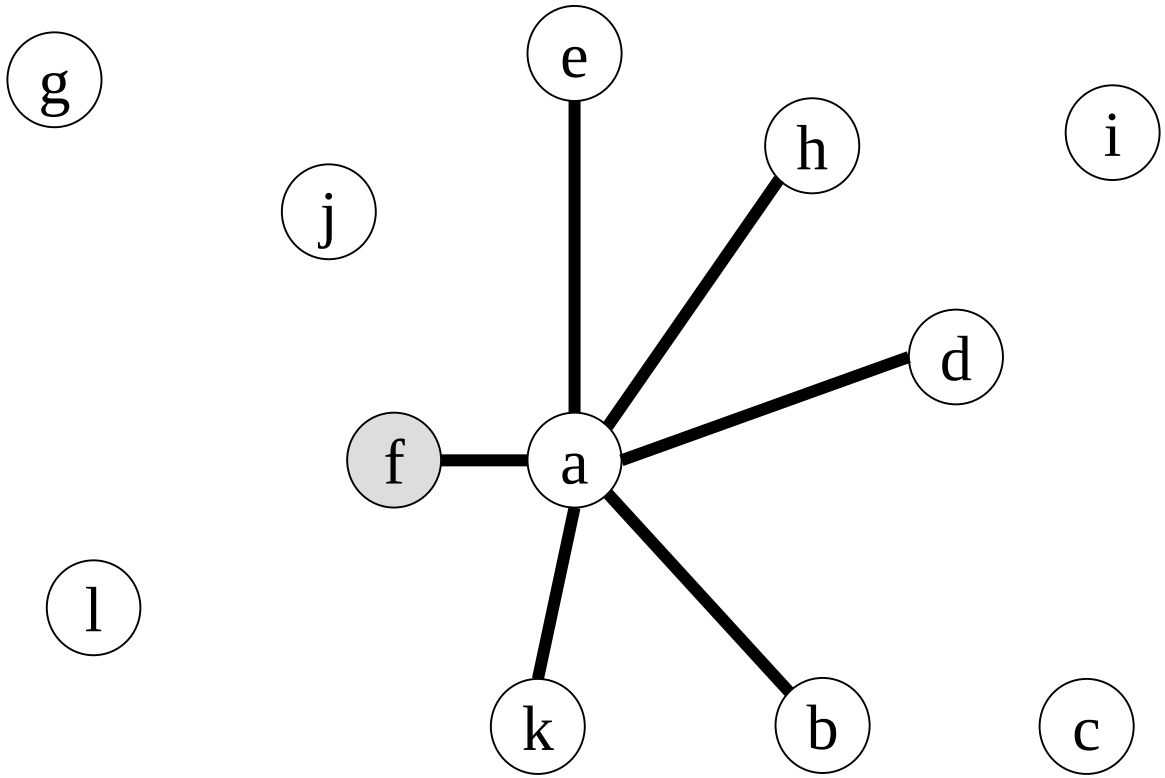



d(Pi, Pj)< d(Pi, Pl)

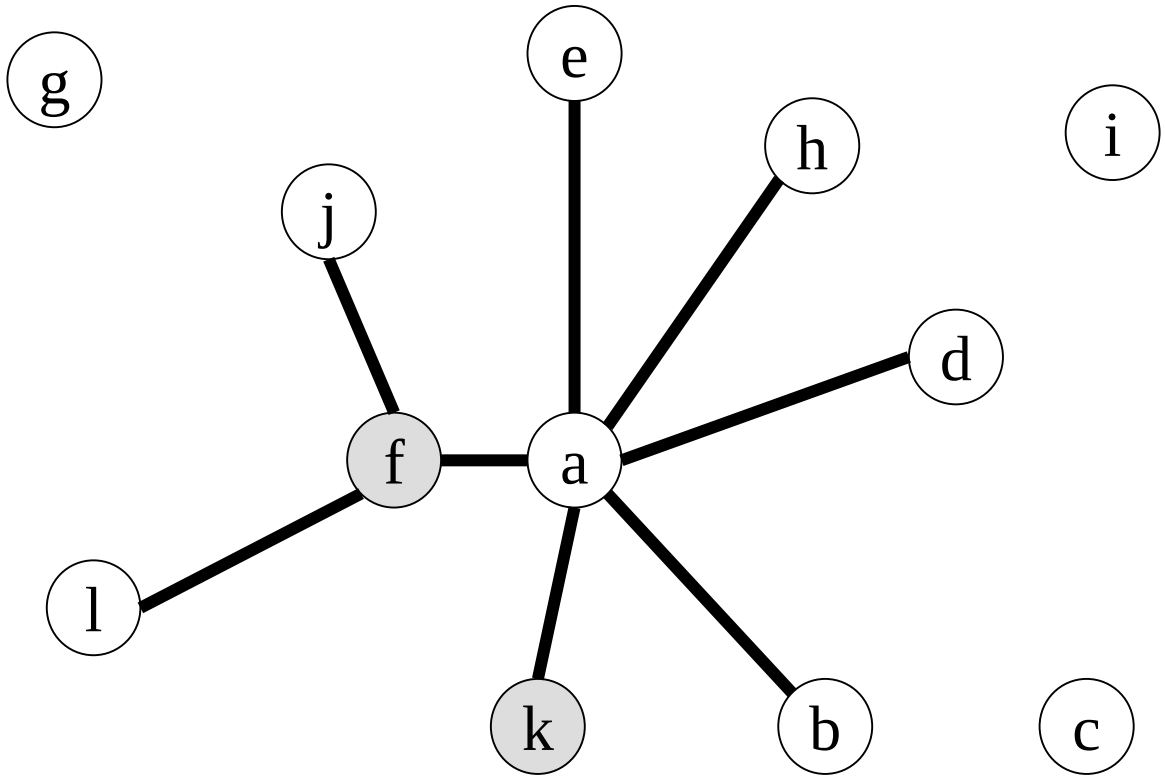
この関係を利用することによって，列挙法による
k nearest neighborsと同じものを，高速に求めることができる



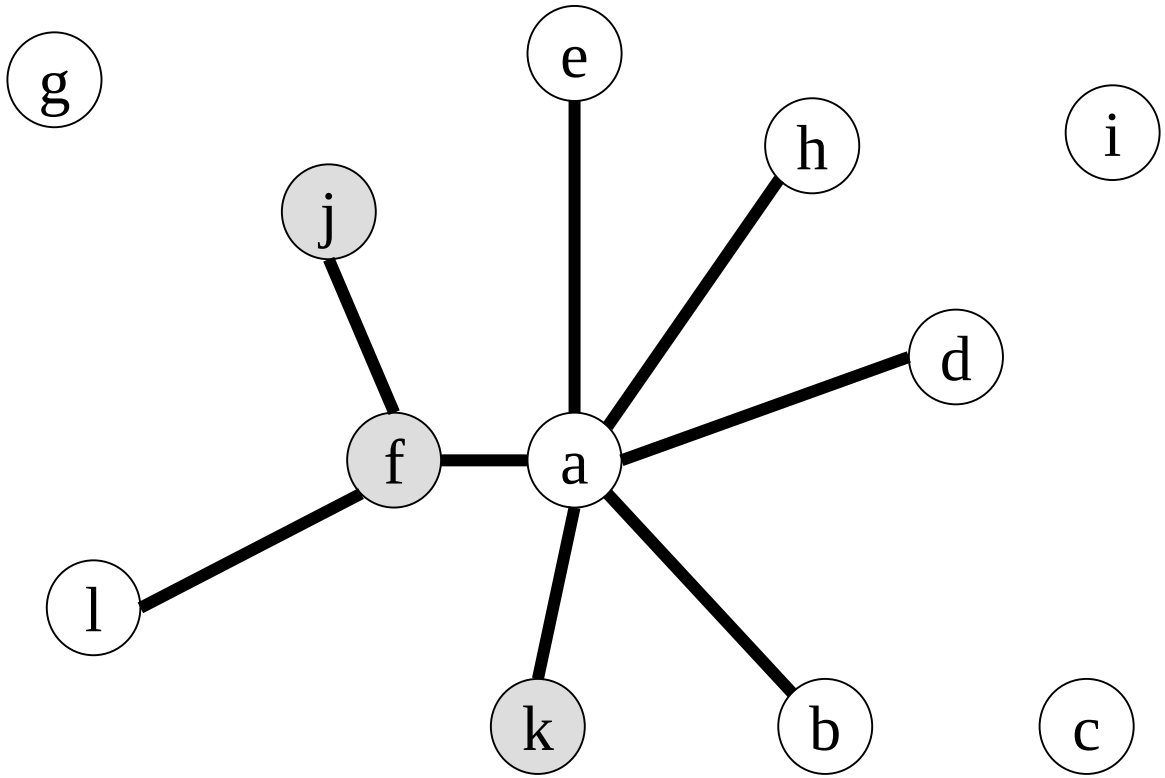

都市aの４ nearest neighborsを求める



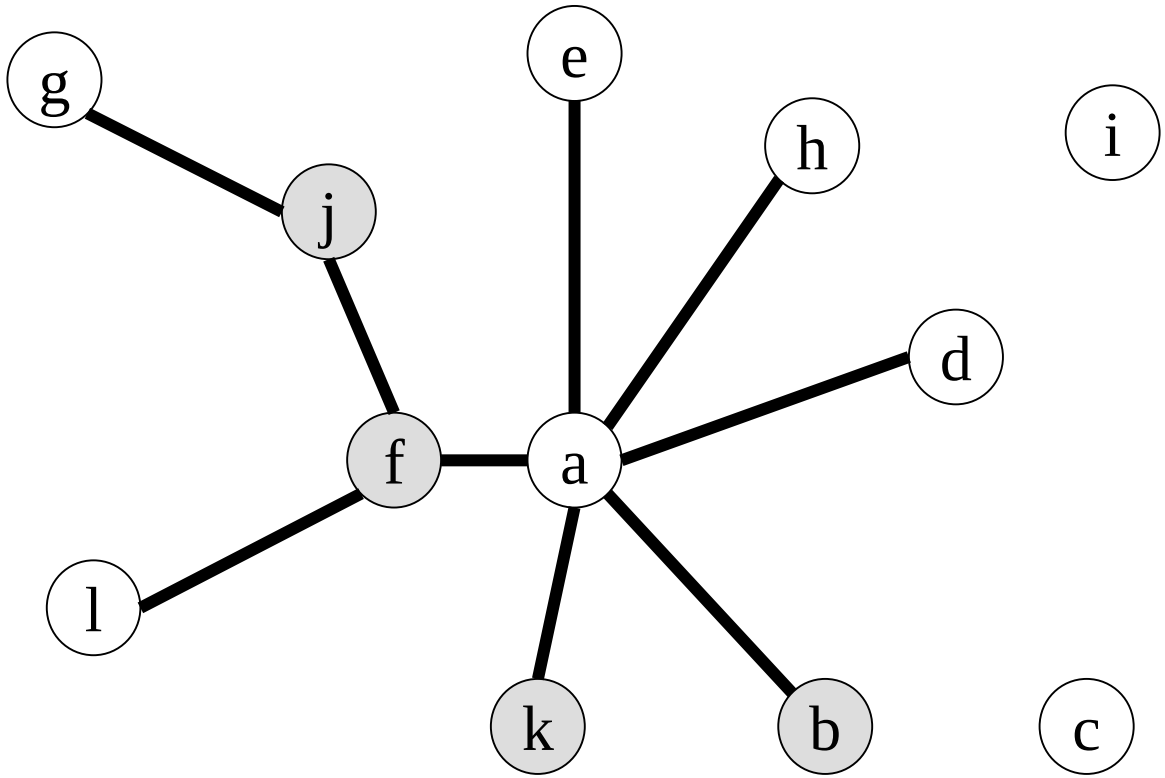

都市 a にドロネーグラフにおいて接続している都市を都市の集合Lとし，Lの中で都市 a に最も近い都市を求める。この場合，都市 f である．Lから都市 f を取り除く



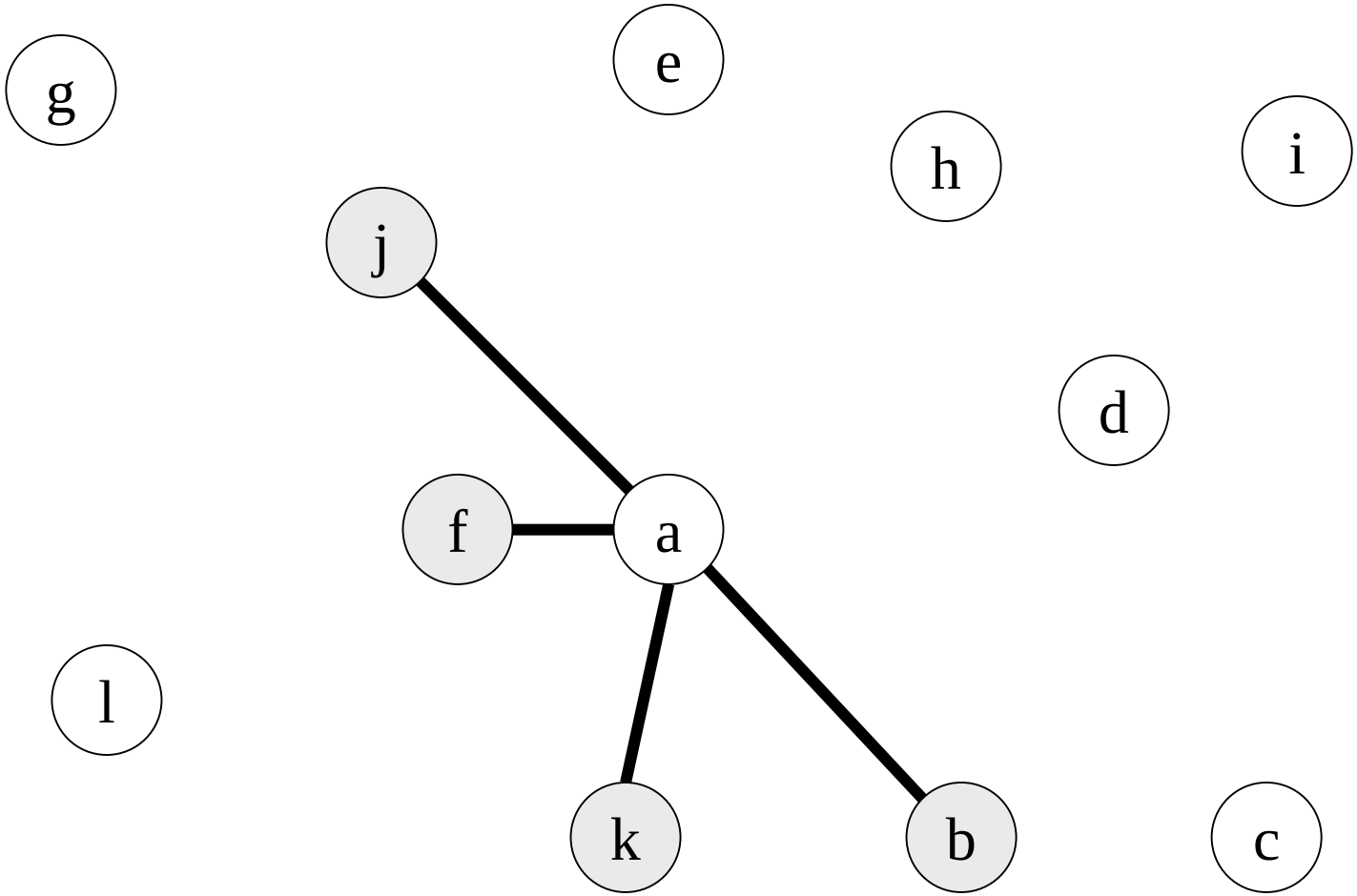

都市ｆにドロネーグラフにおいて接続している都市ｊ，ｌをLに加える。枝ａｊ，ａｌは都市ｆのボロノイ領域を横切っているので，枝ａｆよりも必ず長い．Lの中で都市ａに最も近い都市を求める。この場合都市ｋである．Lから都市ｋを取り除く




新たにLに加える都市はないので，この中で都市ａに最も近い都市を求める。この場合，都市ｊである．
Lから都市ｊを取り除く



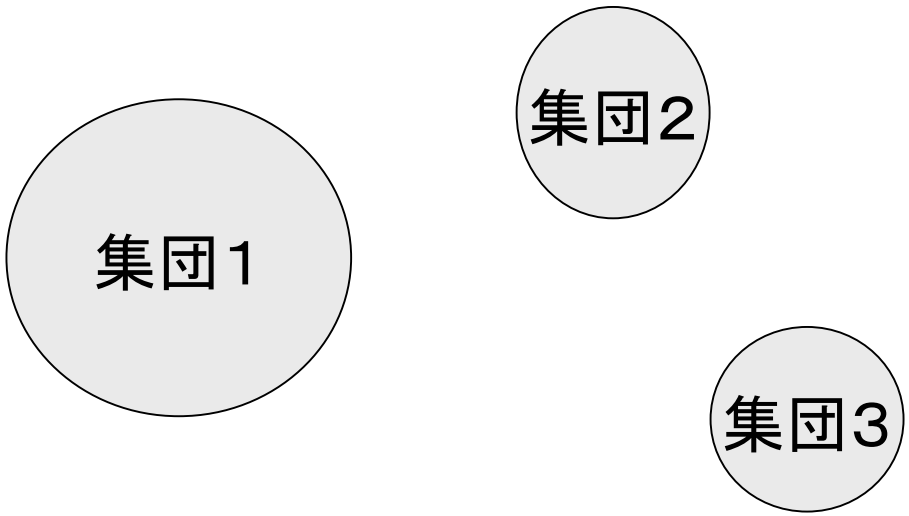

都市ｊにドロネーグラフにおいて接続している都市ｇをLに加える。枝ａｇは都市ｊのボロノイ領域を横切っているので，枝ａｊよりも必ず長い。Lの中で都市ａに最も近い都市を求める．この場合都市ｂである。




都市ｂ，ｆ，ｊ，ｋが都市ａの４ nearest neighborsであり、高速に求めることができる

# k nearest neighborsに基づくCandidate Setの問題点

都市の配置がいくつかの集団に別れている場合




k nearest neighborsが，各集団内で構成されCandidate Setが非連結となる

→ 方法[2]で解決

## ２．４ 方法[２] Delaunay Candidate Set

ドロネーグラフは連結グラフなので、その枝をすべて採用すれば方法［１］の欠点は解決できる

↓

ドロネーグラフの枝をまずCandidate Setに選ぶ

↓

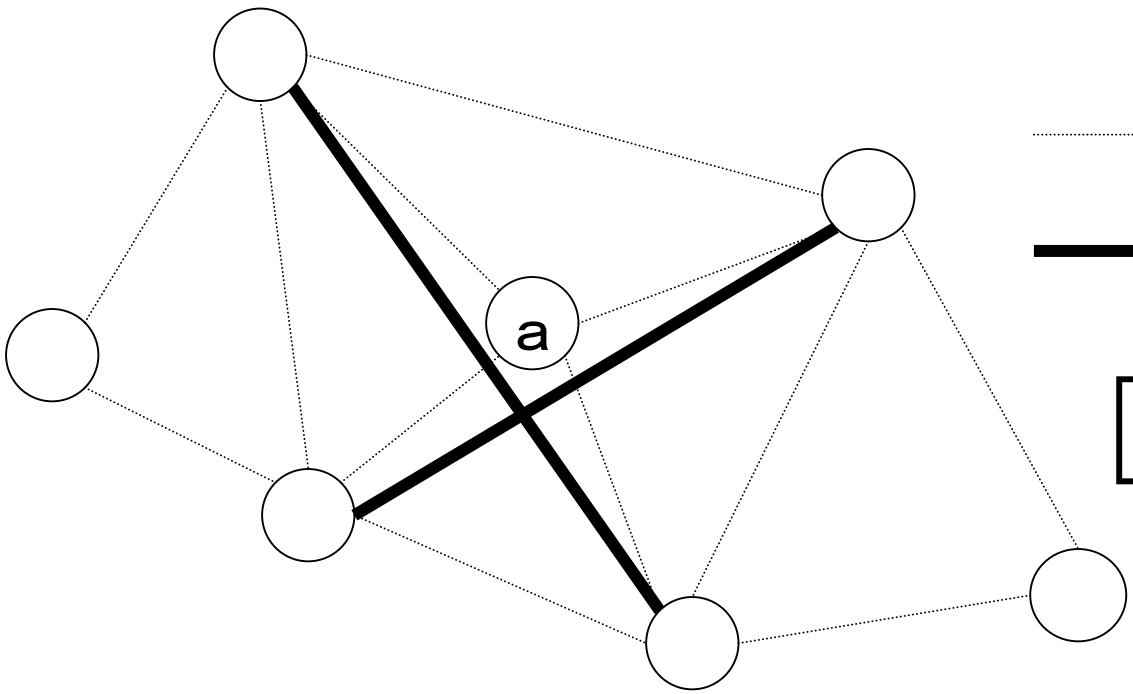
ドロネーグラフの枝のみではCandidate Setとしては、枝の数が少なく十分ではない

↓

次の手順を用い枝を追加する

## 手順

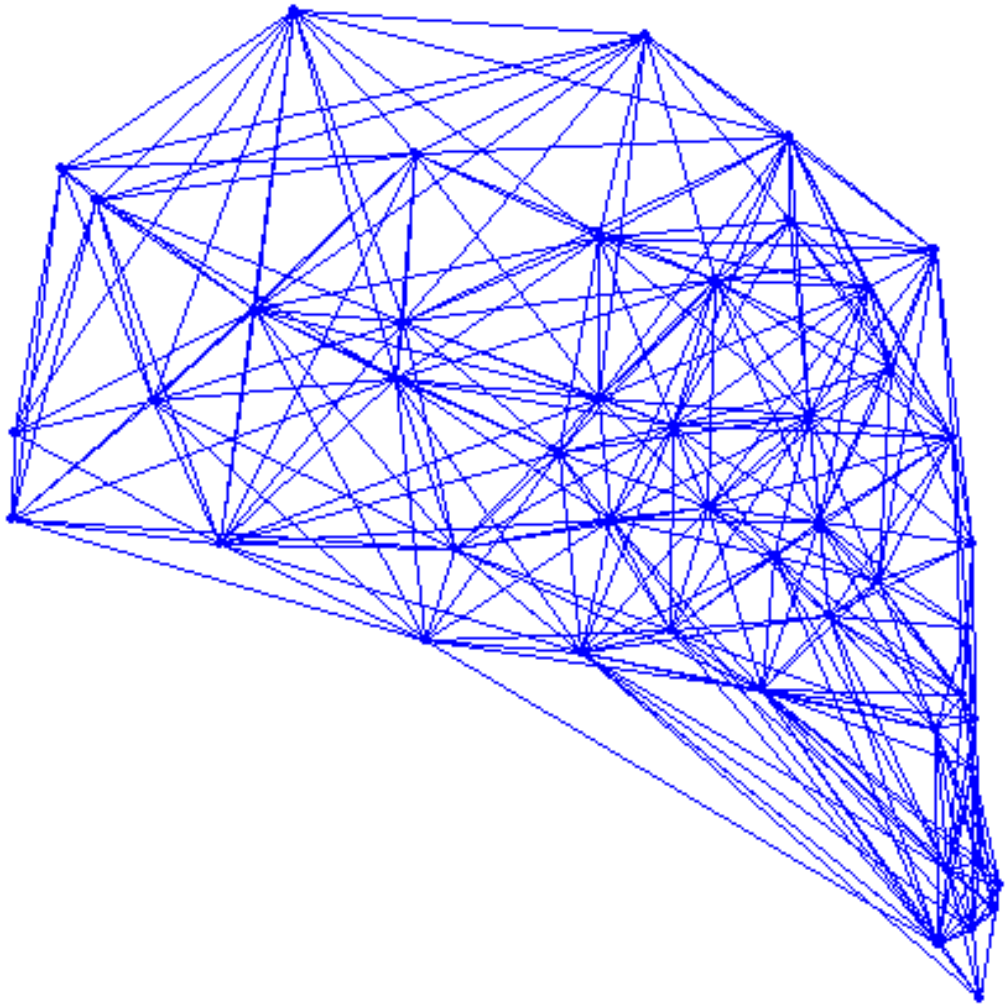
都市Ｐｉ（ｉ＝１,･･･,ｎ）にドロネーグラフにおいて接続している２都市Ｐｊ，Ｐｋにについて枝ＰｊＰｋがCandidate Setに含まれていなければ、その枝をCandidate Setに加える。




ドロネーグラフ

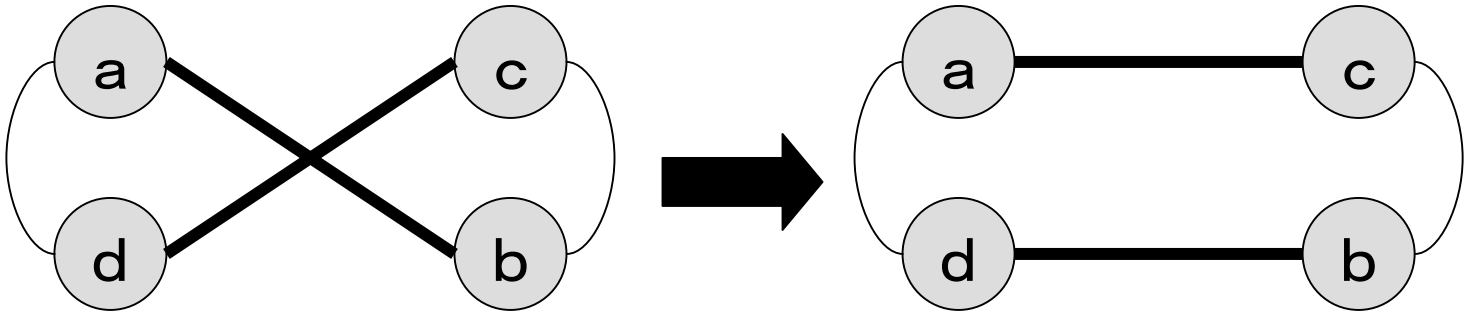
加える枝

都市aについての操作



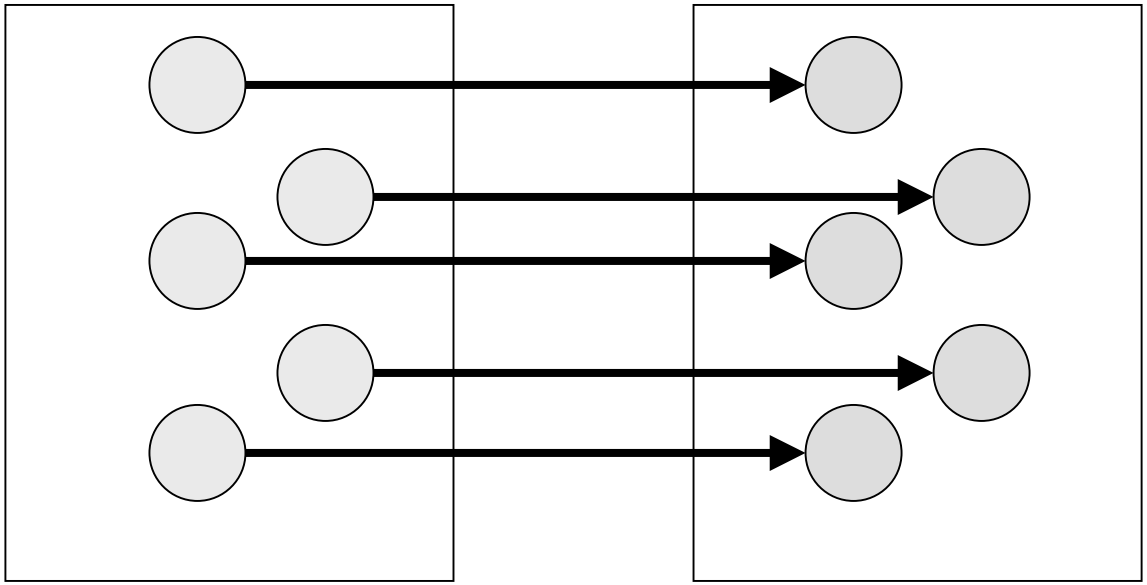
Candidate Set（方法[２]）

# ３．２－opt法




枝ab，cdを除いて，枝ac，bdを加えたとき巡回路の長さが減少していれば交換する

Candidate Set上の２－opt 法では，枝ac，bdがCandidate Setに含まれる場合のみ，枝の交換を考える



初期解　　近似解

ランダムに多数の初期巡回路を作り，２－opt法を用いる

# 4. 実験結果

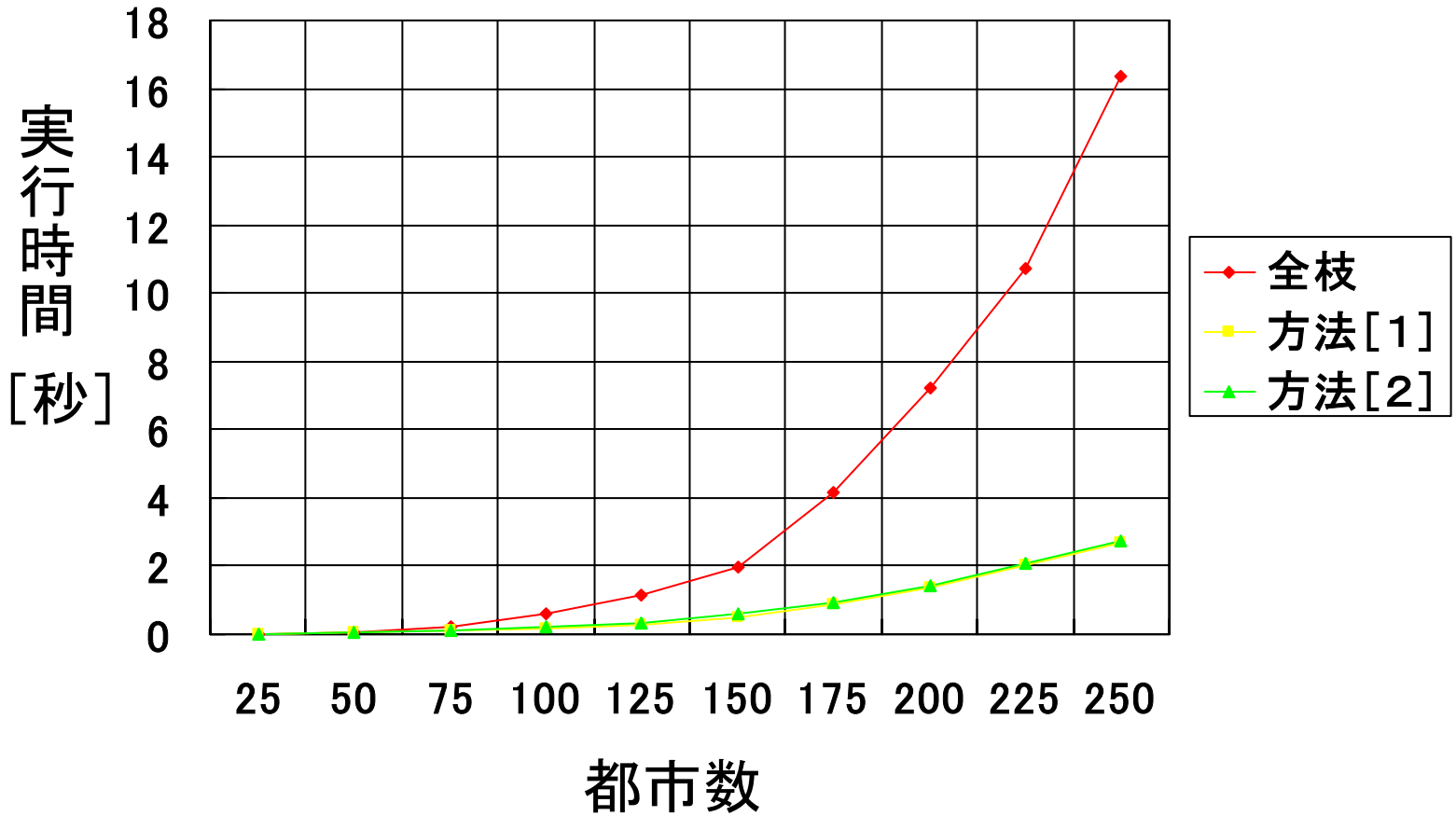

実行時間［秒］
18
16
14
12
10
8
6
4
2
0
25
50
75
100
125
150
175
200
225
250
都市数
全枝
方法[1]
方法[2]

表1．att48問題の結果（都市数48，最適解10628）

| | 時間［秒］(平均) | 準最適解(平均) | 準最適解(平均)／最適解 | 最良の解 | 枝数 |
|---|---|---|---|---|---|
| 全枝 | ０．０７５０ | １１０５２．８ | １．０３９９ | １０６２８ | １１２８ |
| 方法［1］（ｋ＝８） | ０．０３１１ | １１１５５．１ | １．０４９６ | １０６４８ | ２３０ |
| 方法［1］（ｋ＝１０） | ０．０３１９ | １１１３３．７ | １．０４７６ | １０６２８ | ２８２ |
| 方法［1］（ｋ＝１２） | ０．０３６９ | １１０７５．３ | １．０４２１ | １０６２８ | ３４３ |
| 方法［2］ | ０．０３４７ | １１２２６．０ | １．０５６３ | １０６３８ | ３２５ |

表2．berlin52問題の結果（都市数52，最適解7542）

| | 時間［秒］(平均) | 準最適解(平均) | 準最適解(平均)／最適解 | 最良の解 | 枝数 |
|---|---|---|---|---|---|
| 全枝 | ０．０７７８ | ８１３５．２ | １．０７８７ | ７５４２ | １３２６ |
| 方法［1］（ｋ＝８） | ０．０３３６ | ８４１６．５ | １．１１５９ | ７６６３ | ２８１ |
| 方法［1］（ｋ＝１０） | ０．０３９９ | ８３５６．０ | １．１０７９ | ７５４２ | ３５２ |
| 方法［1］（ｋ＝１２） | ０．０４３０ | ８２９８．１ | １．１００２ | ７５４２ | ４２７ |
| 方法［2］ | ０．０４０８ | ８３２５．６ | １．１０３８ | ７５４２ | ３９６ |

結果は1000回の探索の平均

# ５．おわりに

## 考察

方法[１]，方法[２]とも，全枝に比べてやや精度は悪くなるが，ほぼ同程度の精度で準最適解を求めることができ，実行時間はかなり減少する

↓

Candidate Setは有効である

方法[１]でのkの選択 ⟶ ｋ＝１０程度が適当

# 問題点

◎ 方法[1]で，ドロネーグラフを利用する事により，高速にk nearest neighborsを求めることができるが，軽減された時間はドロネーグラフを求める時間により相殺される