

行列の例

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & & a_{2N} \\ a_{31} & a_{32} & a_{33} & & a_{3N} \\ \vdots & & & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{pmatrix}$$

$$\begin{aligned}
H = & A_1 \sum_{v \in V} \left(1 - \sum_{j=1}^N x_{v,j} \right)^2 + A_2 \sum_{j=1}^N \left(1 - \sum_{v \in V} x_{v,j} \right)^2 \\
& + A_3 \sum_{(u,v) \notin E} \sum_{j=1}^N x_{u,j} x_{v,j+1} + A_4 \sum_{(uv) \in E} W_{uv} \sum_{j=1}^N x_{u,j} x_{v,j+1}
\end{aligned}$$

```

const cities := [@NS, @KD, @NI, @CM, @HM, ...]
type order := 1 .. len(cities)
order cities
define distance @A @B := (@A <-> @B || (@A == 1,
    @B == len(cities)) || (@A == len(cities), @B == 1))
different (cities)
minimize {
    41.1 : distance @KS @KD
    119 : distance @KS @NI
    74.2 : distance @KS @CM
    139 : distance @KS @HM
    86.4 : distance @KD @NI
    37.9 : distance @KD @CM
    110 : distance @KD @HM
    67.4 : distance @NI @CM
    121 : distance @NI @HM
    81.1 : distance @CM @HM ...
}
solve(cities)
print (cities)

```

```
const cities := [@NS, @KD, @NI, @CM, @HM, ...]
```

```
type order := 1 .. len(cities)
```

```
order cities
```

```
define distance @A @B := (@A <-> @B || (@A == 1,  
    @B == len(cities)) || (@A == len(cities), @B == 1))
```

```
different (cities)
```

```
minimize {
```

```
    41.1 : distance @KS @KD
```

```
    119 : distance @KS @NI
```

```
    74.2 : distance @KS @CM
```

```
    139 : distance @KS @HM
```

```
    86.4 : distance @KD @NI
```

```
    37.9 : distance @KD @CM
```

```
    110 : distance @KD @HM
```

```
    67.4 : distance @NI @CM
```

```
    121 : distance @NI @HM
```

```
    81.1 : distance @CM @HM ...
```

```
}
```

```
solve(cities)
```

```
print(cities)
```

cities の定義

```
const cities := [@NS, @KD, @NI, @CM, @HM, ...]
```

```
type order := 1 .. len(cities)
```

```
order cities
```

```
define distance @A @B := (@A <-> @B || (@A == 1,  
    @B == len(cities)) || (@A == len(cities), @B == 1))
```

```
different (cities)
```

```
minimize {
```

```
    41.1 : distance @KS @KD
```

```
    119 : distance @KS @NI
```

```
    74.2 : distance @KS @CM
```

```
    139 : distance @KS @HM
```

```
    86.4 : distance @KD @NI
```

```
    37.9 : distance @KD @CM
```

```
    110 : distance @KD @HM
```

```
    67.4 : distance @NI @CM
```

```
    121 : distance @NI @HM
```

```
    81.1 : distance @CM @HM ...
```

```
}
```

```
solve(cities)
```

```
print(cities)
```

cities の定義

distance の
定義

```
const cities := [@NS, @KD, @NI, @CM, @HM, ...]
```

```
type order := 1 .. len(cities)
```

```
order cities
```

```
define distance @A @B := (@A <-> @B || (@A == 1,  
    @B == len(cities))) || (@A == len(cities), @B == 1))
```

```
different (cities)
```

```
minimize {
```

```
41.1 : distance @KS @KD
```

```
119 : distance @KS @NI
```

```
74.2 : distance @KS @CM
```

```
139 : distance @KS @HM
```

```
86.4 : distance @KD @NI
```

```
37.9 : distance @KD @CM
```

```
110 : distance @KD @HM
```

```
67.4 : distance @NI @CM
```

```
121 : distance @NI @HM
```

```
81.1 : distance @CM @HM ...
```

```
}
```

```
solve(cities)
```

```
print(cities)
```

cities の定義

distance の
定義

最小化

```
const cities := [@NS, @KD, @NI, @CM, @HM, ...]
```

```
type order := 1 .. len(cities)
```

```
order cities
```

```
define distance @A @B := (@A <-> @B || (@A == 1,  
    @B == len(cities)) || (@A == len(cities), @B == 1))
```

```
different (cities)
```

```
minimize {
```

```
41.1 : distance @KS @KD
```

```
119 : distance @KS @NI
```

```
74.2 : distance @KS @CM
```

```
139 : distance @KS @HM
```

```
86.4 : distance @KD @NI
```

```
37.9 : distance @KD @CM
```

```
110 : distance @KD @HM
```

```
67.4 : distance @NI @CM
```

```
121 : distance @NI @HM
```

```
81.1 : distance @CM @HM ...
```

```
}
```

```
solve(cities)
```

```
print(cities)
```

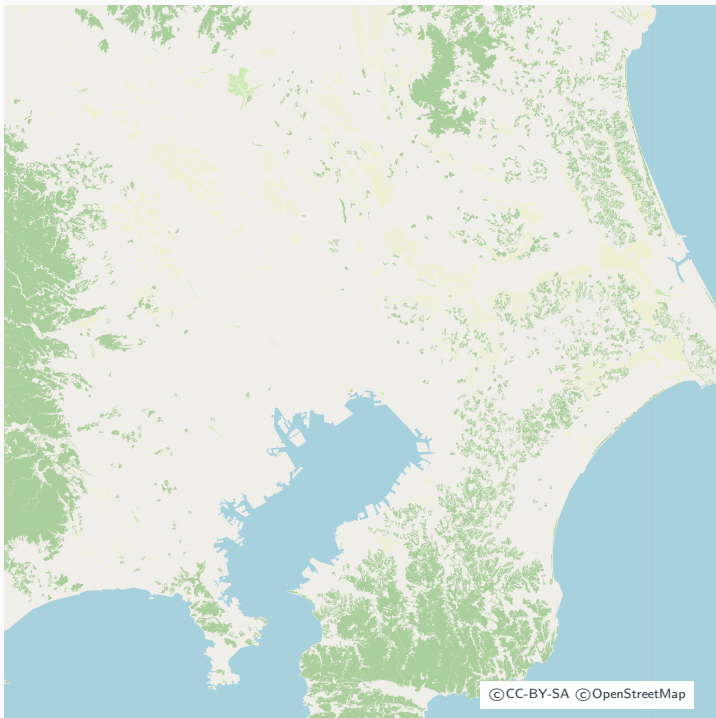
cities の定義

distance の
定義

最小化

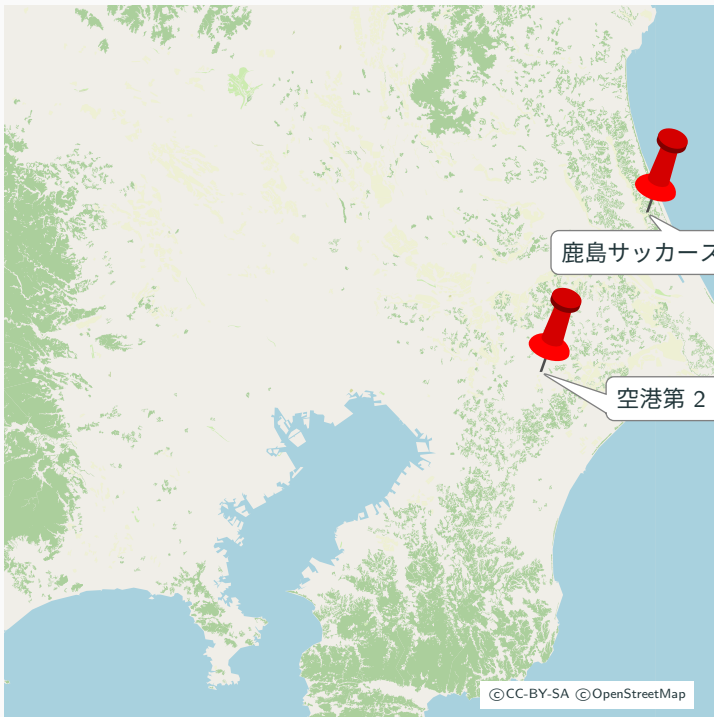
なんとなく分かる

高輪ゲートウェイ



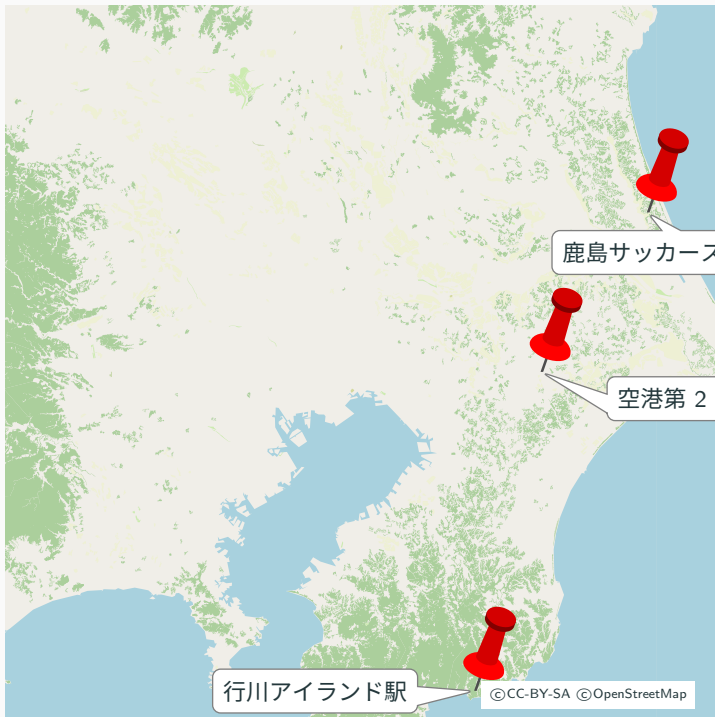


鹿島サッカースタジアム駅



鹿島サッカースタジアム駅

空港第2ビル駅



鹿島サッカースタジアム駅

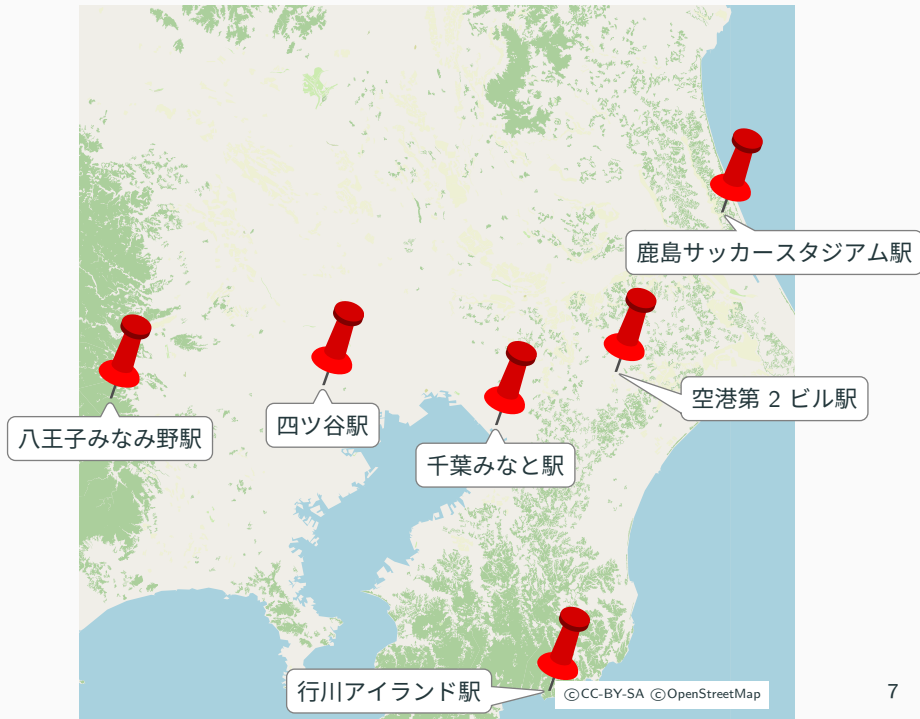
空港第2ビル駅

行川アイランド駅

©CC-BY-SA ©OpenStreetMap











あしががフラワーパーク駅

さいたま新都心駅

鹿島サッカースタジアム駅

八王子みなみ野駅

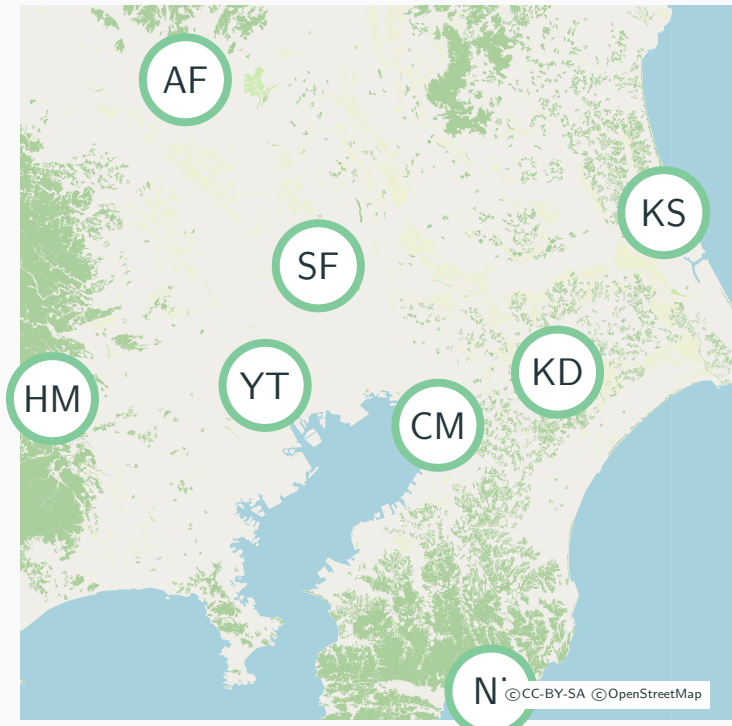
四ツ谷駅

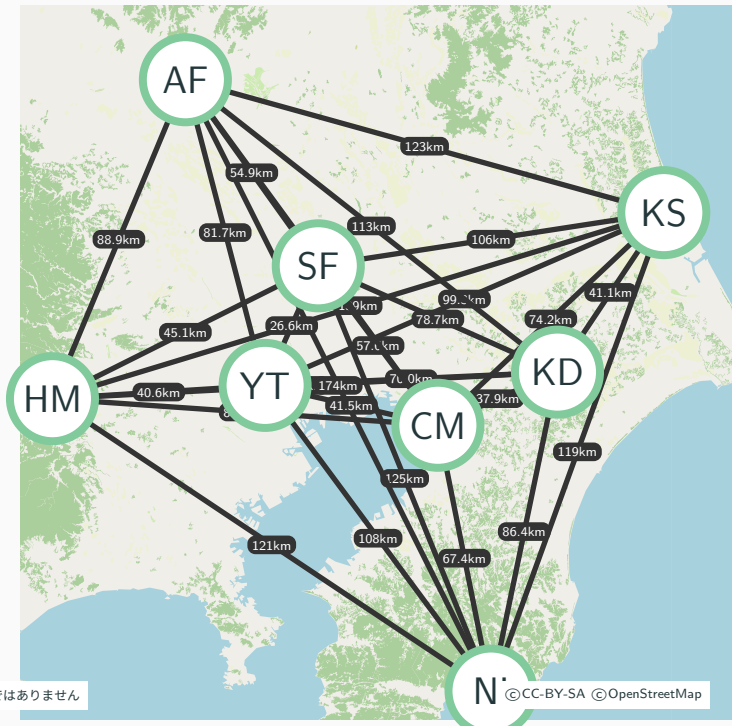
空港第2ビル駅

千葉みなと駅

行川アイランド駅


©CC-BY-SA ©OpenStreetMap






※道のりは正確ではありません

巡回セールスマン問題とは


すべての  を 1 度ずつ訪問し戻ってくるルートの中で
道のりが最短なものを 求める。

巡回セールスマン問題とは

すべての  を1度ずつ訪問し戻ってくるルートの中で
制約条件

道のりが最短なものを求める。
最適化項

巡回セールスマン問題とは


すべての  を1度ずつ訪問し戻ってくるルートの中で
制約条件

道のりが最短なものを求める。
最適化項

制約条件 + 最適化項 =



巡回セールスマン問題とは

すべての  を1度ずつ訪問し戻ってくるルートの中で
制約条件

道のりが最短なものを求める。
最適化項

制約条件 + 最適化項 = 組合せ最適化問題


```
const cities := [@NS, @KD, @NI, @CM, @HM, ...]
```

```
type order := 1 .. len(cities)
```

```
order cities
```

```
define distance @A @B := (@A <-> @B || (@A == 1,  
    @B == len(cities)) || (@A == len(cities), @B == 1))
```

```
different (cities)
```

```
minimize {
```

```
41.1 : distance @KS @KD
```

```
119 : distance @KS @NI
```

```
74.2 : distance @KS @CM
```

```
139 : distance @KS @HM
```

```
86.4 : distance @KD @NI
```

```
37.9 : distance @KD @CM
```

```
110 : distance @KD @HM
```

```
67.4 : distance @NI @CM
```

```
121 : distance @NI @HM
```

```
81.1 : distance @CM @HM ...
```

```
}
```

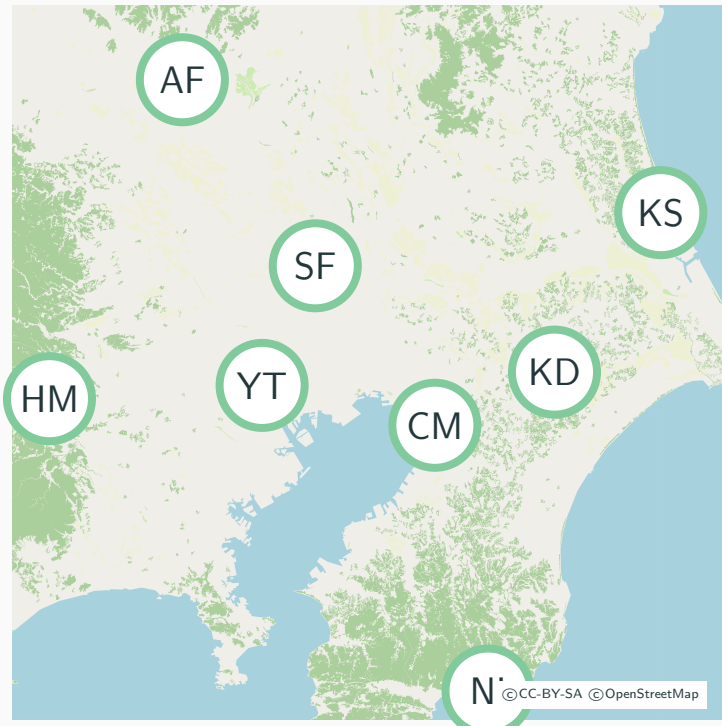
```
solve(cities)
```

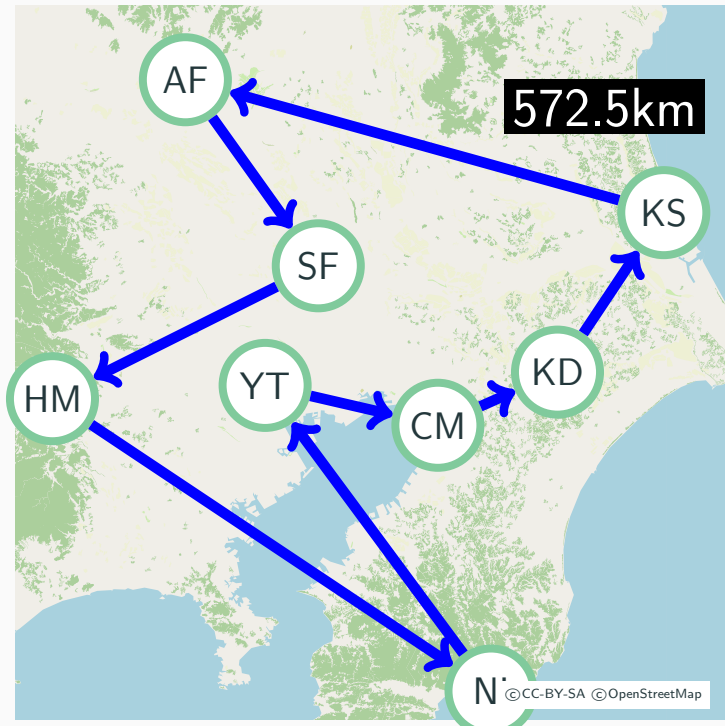
```
print(cities)
```

cities の定義

distance の
定義

最小化





組合せ最適化問題

世の中の多くの問題は
組合せ最適化問題 で記述される



アニーリング によって
組合せ最適化問題を早く解けるようになると期待されている

他の組合せ最適化問題の例

- グラフ分割問題
- 集合詰め問題
- 頂点被覆問題
- 最小最大マッチング問題
- グラフ彩色問題
- ハミルトン閉路問題
- ...

他の組合せ最適化問題の例

- グラフ分割問題
- 集合詰め問題
- 頂点被覆問題
- 最小最大マッチング問題
- グラフ彩色問題
- ハミルトン閉路問題
- ...

```
type group A | B | C
group [ @NODE_1, @NODE_2, @NODE_3, @NODE_4,
        @NODE_5, @NODE_6, @NODE_7, @NODE_8 ] : p
minimize {
  1: @NODE_1 == @NODE_2
  1: @NODE_1 == @NODE_3
  1: @NODE_1 == @NODE_5
  1: @NODE_1 == @NODE_6
  1: @NODE_1 == @NODE_8
  1: @NODE_2 == @NODE_4
  1: @NODE_2 == @NODE_5
  1: @NODE_2 == @NODE_7
  1: @NODE_3 == @NODE_4
  1: @NODE_3 == @NODE_5
  1: @NODE_3 == @NODE_6
  1: @NODE_3 == @NODE_8
  1: @NODE_4 == @NODE_5
  1: @NODE_4 == @NODE_7
  1: @NODE_5 == @NODE_7
  1: @NODE_6 == @NODE_7
}
solve(p)
print(p)
```

他の組合せ最適化問題の例

- グラフ分割問題
- 集合詰め問題
- 頂点被覆問題
- 最小最大マッチング問題
- グラフ彩色問題
- ハミルトン閉路問題
- ...

```
type group A | B | C
group [ @NODE_1, @NODE_2, @NODE_3, @NODE_4,
        @NODE_5, @NODE_6, @NODE_7, @NODE_8 ] : p
```

```
type bool := yes | no
bool [ @v1, @v2, @v3, @v4, @v5,
        @v6, @v7, @v8 ] : subsets

@v1 != yes || @v4 != yes
@v1 != yes || @v6 != yes
@v2 != yes || @v5 != yes
@v2 != yes || @v8 != yes
@v3 != yes || @v7 != yes
@v4 != yes || @v5 != yes
@v4 != yes || @v8 != yes
@v5 != yes || @v7 != yes
@v7 != yes || @v8 != yes

maximize {
  1 : @v1 == yes
  1 : @v2 == yes
  1 : @v3 == yes
  1 : @v4 == yes
  1 : @v5 == yes
  1 : @v6 == yes
  1 : @v7 == yes
  1 : @v8 == yes
}
solve(subsets)
print(subsets)
```

他の組合せ最適化問題の例

- グラフ分割問題
- 集合詰め問題
- 頂点被覆問題
- 最小最大マッチング問題
- グラフ彩色問題
- ハミルトン閉路問題
- ...

```
type group A | B | C
group [ @NODE_1, @NODE_2, @NODE_3, @NODE_4,
        @NODE_5, @NODE_6, @NODE_7, @NODE_8 ] : p
```

```
type bool := yes | no
bool [ @v1, @v2, @v3, @v4, @v5,
       @v6, @v7, @v8 ] : subsets
```

```
@v1 != yes || @v4 != yes
@v1 != yes || @v6 != yes
```

@v

@v

@v

@M

@v

④

@v
@v

100

14

1.

134

13

13

13

13

13

13

}

solv

prim

```
type colored := yes | no
colored [ @v1, @v2, @v3, @v4, @v5,
          @v6, @v7, @v8 ] : vertices
@v1 == yes, @v4 == yes
@v2 == yes, @v5 == yes
@v4 == yes, @v5 == yes
@v5 == yes, @v7 == yes
```

```

minimize {
  1:@v1 == yes
  1:@v2 == yes
  1:@v3 == yes
  1:@v4 == yes
  1:@v5 == yes
  1:@v6 == yes
  1:@v7 == yes
  1:@v8 == yes
}
solve(vertices)
print(vertices)

```


他の組合せ最適化問題の例

- グラフ分割問題
- 集合詰め問題
- 頂点被覆問題
- 最小最大マッチング問題
- グラフ彩色問題
- ハミルトン閉路問題
- ...

```
type group A | B | C
group [ @NODE_1, @NODE_2, @NODE_3, @NODE_4,
        @NODE_5, @NODE_6, @NODE_7, @NODE_8 ] : p
```

```
type bool := yes | no
bool [ @v1, @v2, @v3, @v4, @v5,
        @v6, @v7, @v8 ] : subsets
@v1 != yes || @v4 != yes
@v1 != yes || @v6 != yes
@v...
```

```
type colored := yes | no
colored [ @v1, @v2, @v3, @v4, @v5,
           @v6, @v7, @v8 ] : vertices
@v1 == yes, @v4 == yes
@v2 == yes, @v5 == yes
```

```
type color red | orange | pink
color [ @BC, @YK, @NW, @AB, @SK, @NV, @MT,
         @ON, @QB, @NB, @NS, @PE, @NL ] : p
@BC != @AB, @BC != @YK, @BC != @NW
@YK != @BC, @YK != @NW
@NW != @YK, @NW != @BC
@AB != @BC, @AB != @NW, @AB != @SK
@SK != @AB, @SK != @NW, @SK != @MT
@NV != @NW, @NV != @MT
@MT != @NV, @MT != @SK, @MT != @ON
@ON != @MT, @ON != @QB
@QB != @ON, @QB != @NB, @QB != @NL
@NB != @QB, @NB != @NS
@NS != @NB
@NL != @QB
solve(p)
print(p)
```


古典コンピュータとアニーリングマシン

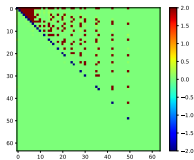
ノイマン型
コンピュータ



```
ADD %rbp  
EOR %rax
```



アニーリング



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

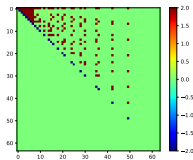
機械語



```
ADD %rbp  
EOR %rax
```



アニーリング



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

機械語

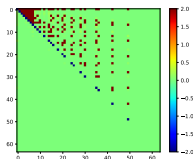


アセンブリ
言語

```
ADD %rbp  
EOR %rax
```



アニーリング



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

機械語



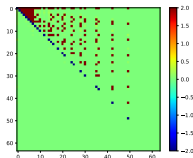
アセンブリ
言語

```
ADD %rbp  
EOR %rax
```

構造化プログ
ラミング言語



アニーリング



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

機械語



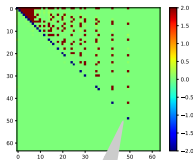
アセンブリ
言語

```
ADD %rbp  
EOR %rax
```

構造化プログ
ラミング言語



アニーリング



QUBO



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

機械語



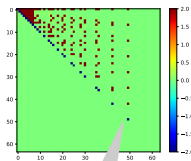
アセンブリ
言語

```
ADD %rbp  
EOR %rax
```

構造化プログ
ラミング言語



アニーリング



QUBO



ライブラリ
(PyQUBO など)



古典コンピュータとアニーリングマシン

ノイマン型
コンピュータ

機械語



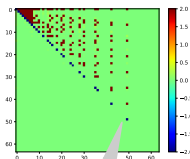
アセンブリ
言語

```
ADD %rbp  
EOR %rax
```

構造化プログ
ラミング言語



アニーリング



QUBO



ライブラリ
(PyQUBO など)




??

SpoonQ の紹介

A programming language for annealing machine.

[commits](#) [1 branch](#) [0 packages](#) [0 releases](#) [1 contributor](#)

[Branch: ▾](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

 [yassuo-0204](#)

[resources](#)

[src](#)

[test](#)

[.gitignore](#)


[Cargo.lock](#)

[Cargo.toml](#)

[README.md](#)

[rustfmt.toml](#)

[README.md](#)



SpoonQ

SpoonQ

A programming language for annealing machine.

SpoonQ の紹介

A programming language for annealing machine.

commits 1 branch 0 packages 0 releases 1 contributor

Branch: New pull request Create new file Upload files Find file Clone or download

yassuo-0204

resources

src

test

.gitignore

Cargo.lock

Cargo.toml

README.md

rustfmt.toml

README.md



SpoonQ

A programming language for annealing machine.



<https://github.com/SpoonQ/>

SpoonQ の紹介

SpoonQ Wiki

Introduction

SpoonQ is a language to describe problems and solve them on quantum annealer.
[Japanese](#)

Syntax

- [Commands](#)
- [Operators](#)

Examples

- [Number Partitioning](#)
- [Graph Partitioning](#)
- [Cliques](#) - pending
- [Binary Integer Linear Programming](#) - pending
- [Exact Cover](#) - pending
- [Set Packing](#)
- [Vertex Cover](#)
- [Minimal Maximal Matching](#)
- [Set Cover](#) - pending
- [Knapsack with Integer Weights](#) - pending
- [Graph Coloring](#)
- [Clique Cover](#) - pending
- [Job Sequencing](#) - pending
- [Hamilton Cycles and Paths](#)

▼ Pages **15**

[Home](#)

[Commands](#)

[...](#)



SpoonQ の紹介

SpoonQ Wiki

Introduction

... them on quantum annealer.

★ Star

0

Examples

- [Number Partitioning](#)
- [Graph Partitioning](#)
- [Cliques](#) - pending
- [Binary Integer Linear Programming](#) - pending
- [Exact Cover](#) - pending
- [Set Packing](#)
- [Vertex Cover](#)
- [Minimal Maximal Matching](#)
- [Set Cover](#) - pending
- [Knapsack with Integer Weights](#) - pending
- [Graph Coloring](#)
- [Clique Cover](#) - pending
- [Job Sequencing](#) - pending
- [Hamilton Cycles and Paths](#)


▼ Pages 15

Find a Page...

[Home](#)

[Commands](#)

...



SpoonQ の紹介

SpoonQ Wiki

Introduction

... them on quantum annealer.

★ Star

0

Examples

- [Number Partitioning](#)
- [Graph Partitioning](#)

👁 Watch ▼

0

- Knapsack with Integer Weights - pending
- [Graph Coloring](#)
- Clique Cover - pending
- Job Sequencing - pending
- [Hamilton Cycles and Paths](#)

▼ Pages 15

Find a Page...

[Home](#)

[Commands](#)



SpoonQ の紹介

1 問題を記述する



2 答を教えてくれる

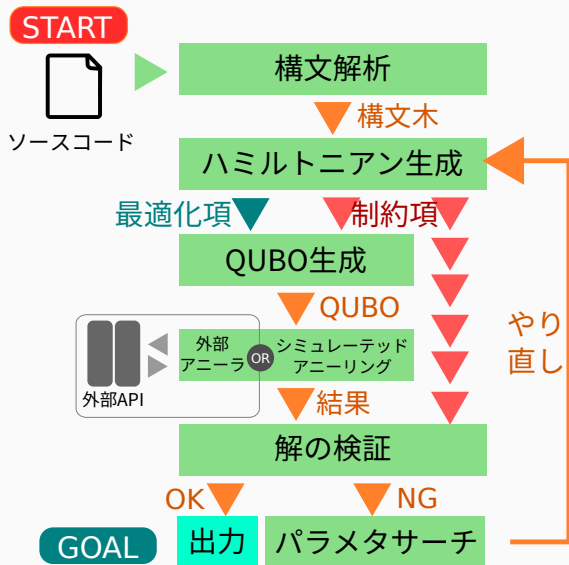


無料ですぐ使える

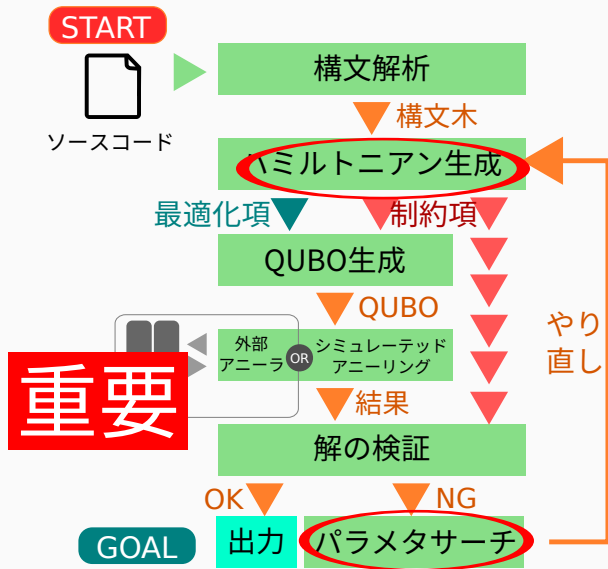
デモ

プログラミングに詳しい方向けの説明

開発者の方へ



開発者の方へ



※現時点で実装されていない機能もあります

現状の SpoonQ では大きな問題は解けない

現状の SpoonQ では大きな問題は解けない



ハミルトニアン生成


パラメタサーチ

に改善の余地


アニメーションへの敷居を下げることで利用人口を増やし...



アニメーションへの敷居を下げることで利用人口を増やし...

2020 年を  東京オリンピック に!

アニーリングへの敷居を下げることで利用人口を増やし...

2020 年を  ~~東京オリンピック~~ に!
アニーリング元年

201X 年は機械学習元年？

201X 年は機械学習元年?

- 機械学習の初期 ... 研究者が独自にアルゴリズムを開発
 - 限られた人しか機械学習を活用できない
- 機械学習ライブラリの発展
 - 利用人口の増加
 - 新しいパラダイムの出現
 - さらなる機械学習ライブラリの出現

2020 年現在

201X 年は機械学習元年?

- 機械学習の初期 ... 研究者が独自にアルゴリズムを開発
 - 限られた人しか 機械学習を 活用できない
- 機械学習ライブラリの発展
 - 利用人口の増加
 - 新しいパラダイムの出現
 - さらなる機械学習ライブラリの出現


アニーリングは
この段階

2020 年現在

201X 年は機械学習元年?

- 機械学習の初期 ... 研究者が独自にアルゴリズムを開発
 - 限られた人しか 機械学習を 活用できない
- 機械学習ライブラリの発展
 - 利用人口の増加
 - 新しいパラダイムの出現
 - さらなる機械学習ライブラリの出現

アニーリングは
この段階

アニーリング
SpoonQ で 業界 に  をつけたい

問 1.

処理系をつくることの困難さ

\equiv

Programming Language

is one of

問 1.

処理系をつくることの困難さ

≡ 問題を記述することの難しさ

Programming Language

is one of

問 1.

処理系をつくることの困難さ

≡ 問題を記述することの難しさ

Programming Language

is one of 思考のための道具



新しい道具をつくり,初めて見える世界を探検したい

まとめ

- SpoonQ はアニーリング向けプログラミング言語
- 有名な組合せ最適化問題だけでなく、より一般的な問題を記述できることを目指している
- ユーザーの方へ アニーリングをもっと身近に!
- 開発者の方へ 「道具」をつくることでアニーリングの発展に貢献!



第 41 回量子情報
技術研究会
(QIT41)

口頭発表



一般社団法人
情報処理学会
Information Processing Society of Japan

情報処理学会
第 82 回全国大会

口頭発表 (予定)



<https://github.com/SpoonQ/>