

PuLP Cheat Sheet

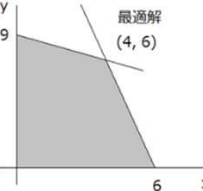
<http://pythonhosted.org/PuLP/>

PuLPとは

最適化問題から数理モデルを作成し、ソルバーで解くためのPythonのライブラリ

基本サンプル

```
from pulp import *
m = LpProblem(sense=LpMaximize)
x = LpVariable('x', lowBound=0)
y = LpVariable('y', lowBound=0)
m += 100 * x + 100 * y
m += x + 2 * y <= 16
m += 3 * x + y <= 18
m.solve()
print(value(x), value(y))
```



数理モデル

```
m = LpProblem() # 最小化
# 最大化
m = LpProblem(sense=LpMaximize)
```

変数

```
LpVariable(名前) # 自由変数
LpVariable(名前, lowBound=0) # 非負変数
LpVariable(名前, cat=LpBinary) # 0-1変数
```

数式

```
2 * x + 3
lpSum([x1, x2, x3]) # 合計
lpDot([1, 2], [x1, x2]) # 内積
```

目的関数

```
m += 目的関数 # 追加ではなく設定
```

制約条件

```
m += 数式または数 == 数式または数
m += 数式または数 >= 数式または数
m += 数式または数 <= 数式または数
```

求解

```
m.solve() # CBC
m.solve(GUROBI_CMD()) # GUROBI
```

結果

```
LpStatus[m.status] # ステータス
value(m.objective) # 目的関数値
value(変数) # 変数値
```

ファイル出力

```
m.writeLP(ファイル名) # LP形式
m.writeMPS(ファイル名) # MPS形式
```

IF条件

```
f(x) > 0 ならば g(x) <= 0
⇔ f(x) <= 0 または g(x) <= 0
⇔ { f(x) <= M * y      (yは0-1変数)
    g(x) <= M * (1-y)
```

最大値の最小化

```
maxi{fi(x)}を最小化
⇔ { 最小化 z (zは自由変数)
    fi(x) <= z for ∀i
```

オプション(計算時間、精度)

```
PULP_CBC_CMD(maxSeconds=1, fracGap=0.01)
```

双対問題(Jupyter Notebookにて)

import dual して

```
%dual
min c^T x
A x >= b
x >= 0
```

主問題	双対問題
$\min c^T x$	$\max b^T y$
$Ax \geq b$	$A^T y \leq c$
$x \geq 0$	$y \geq 0$

輸送最適化問題(Pandas利用)

```
import numpy as np, pandas as pd
from itertools import product
nw, nf = 3, 4 # 倉庫数, 工場数
pr=list(product(range(nw), range(nf)))
供給= np.random.randint(30, 50, nw)
需要= np.random.randint(20, 40, nf)
輸送費=np.random.randint(10, 20, (nw, nf))
a = pd.DataFrame([(i, j) for i, j
                    in pr], columns=['倉庫', '工場'])
a['輸送費'] = 輸送費.flatten()
m = LpProblem()
a['Var'] = [LpVariable('v%d'%i,
                       lowBound=0) for i in a.index]
m += lpDot(a.輸送費, a.Var)
for k, v in a.groupby('倉庫'):
    m += lpSum(v.Var) <= 供給[k]
for k, v in a.groupby('工場'):
    m += lpSum(v.Var) >= 需要[k]
m.solve()
a['Val'] = a.Var.apply(value)
print(a[a.Val > 0])
```

数独(NumPy利用)

```
from ortoolpy import addbinvars
m = LpProblem()
x = np.array(addbinvars(9, 9, 9))
for i, j in product(range(9), range(9)):
    m += lpSum(x[:, i, j]) == 1
    m += lpSum(x[i, :, j]) == 1
    m += lpSum(x[i, j, :]) == 1
    k, l = i//3*3, i%3*3
    m += lpSum(x[k:k+3, l:l+3, j]) == 1
    c = s[i*9+j]
    if str.isnumeric(c):
        m += x[i, j, int(c)-1] == 1
m.solve()
np.vectorize(value)(x).dot(range(1, 10))
```

ORToolPy Cheat Sheet

典型問題

グラフ・ネットワーク問題

最小全域木問題 `nx.minimum_spanning_tree`
最大安定集合問題 `maximum_stable_set`
最大カット問題 `maximum_cut`
最短路問題 `nx.dijkstra_path`
最大流問題 `nx.maximum_flow`
最小費用流問題 `nx.min_cost_flow`

経路問題

運搬経路問題 `vrp`
巡回セールスマン問題 `tsp`

集合被覆・分割問題

集合被覆問題 `set_covering`
集合分割問題 `set_partition`

スケジューリング問題

ジョブショップ問題 `two_machine_flowshop`
勤務スケジューリング問題 `shift_scheduling`

切出し・詰込み問題

ナップサック問題 `knapsack`
ビンパッキング問題 `binpacking`
n次元パッキング問題 `TwoDimPacking`

配置問題

施設配置問題 `facility_location`
容量制約なし施設配置問題 `facility_location_without_capacity`

割当・マッチング問題

2次割当問題 `quad_assign`
一般化割当問題 `gap`
最大マッチング問題 `nx.max_weight_matching`
重みマッチング問題 `nx.max_weight_matching`
安定マッチング問題 `stable_matching`

汎用問題

	連続のみ	離散もOK
線形のみ	線形最適化	混合整数最適化
非線形もOK	非線形最適化	

その他

`logistics_network` ロジスティクスネットワーク問題
`sudoku` 数独

変数生成(PuLP用)

`addvar` 変数作成
`addvars` 多次元変数作成
`addbinvar` 0-1変数作成
`addbinvars` 0-1多次元変数作成

サンプル

```
x = addvar() # 非負変数
x = addvar(lowBound=None) # 自由変数
x = addvars(2,3,4) # 3次元変数
```

区分線形制約(PuLP用)

`addline` 2点直線制約
`addlines` 区分線形制約(非凸)
`addlines_conv` 区分線形制約(凸)

ユーティリティ

`value_or_zero` 評価(未定義なら0)
`graph_from_table` 表からグラフを作成
`networkx_draw` グラフを描画
`L` ユニークなラベル

最適化事例

- オンライン業務における大規模データベースの最適配置(通信業)
- 多重無線ネットワークの最適設計(電力業)
- 移動体通信におけるチャネルの最適割当(通信業)
- 地上デジタルTVにおけるチャネルの最適割当(官公庁)
- 自動車船積付け支援システム設計開発(流通業)
- 光MPLS網の最適パス設計(通信業)
- ロジスティクス最適ネットワーク設計開発(製造業)
- 3次元パッキング最適化システム開発(流通業)
- 緊急物資最適配送計画システム開発(独立行政法人)
- 衛星通信最適リソース割当(通信業)
- ビークル間連携最適配送計画システム開発(官公庁)
- 空箱の輸送コスト最適システム開発(流通業)
- シフトスケジューリングシステム開発(販売業)
- バスの仕業作成最適化(製造業)

