実習２０２５

強化学習編

作成：原　武史（岐阜大学）＋ChatGPT5＋ChatGPT4o-mini
作成日：2025/8/30，2025/9/7修正

目　　次

この書類の目的

いわゆる数理最適化問題は，Operations Research分野で十分に議論されてきた.
ここでは，そのような内容はすっとばして，機械学習（特に強化学習：Reinforcement Learning, RL）を使って解法や順最適解を探索するためのヒントを記す.

配布内容

ReinforcementLearning.zip

にサンプルプログラムをまとめる.

動作環境の構築（別途資料あり）

１．Python環境の構築
　　venvを利用してPythonの仮想環境を構築.

２．仮想環境へのライブラリのインストール
　　以下のPythonライブラリが必要です.
　　仮想環境へpip/pip3でインストールしてください.
　　数値計算系：numpy, scipy,
　　グラフ・表示系：matplotlib, scikit-image, pillow
　　表の処理系：pandas
　　機械学習系：scikit-learn
　　内部処理系：joblib, tqdm
　　強化学習関連：gymnasium

３．同封のプログラムを実行するとバージョンが表示される.
　　以下は原が実行している環境である.

```
$ python3 check_versions.py
numpy       : 1.26.4
scipy       : 1.15.3
matplotlib  : 3.10.3
skimage     : 0.25.2
PIL         : 10.2.0
pandas      : 2.3.0
sklearn     : 1.6.1
joblib      : 1.5.1
tqdm        : 4.67.1
gymnasium   : 1.2.0
```

４．jupyter notebookの起動

（注意）
なし.

迷路の解法探索を強化学習で行う

ファイル名：　　maze.ipynb

目的：強化学習の概念を理解する

方法：迷路の解法探索を強化学習の問題に落とし込む

（概念）
強化学習の基本は MDP（Markov Decision Process，マルコフ決定過程） です.
迷路は自然にこの形に当てはまります：
・状態 (State)：今どのマスにいるか
・行動 (Action)：上下左右の移動
・遷移 (Transition)：行動をとると次のマスに移動（壁があれば移動できない）
・報酬 (Reward)：
　　ゴールに着いたら +1
　　それ以外は 0（または −0.01 で「早くたどり着け」を促す）

（動作原理）
最初はランダムに動くので「迷子」になります.
しかし，ゴールに到達したときだけ報酬を受け取る
すなわち，「この経路がよかった」と経験が残る.
Q学習は，「その状態でその行動を選んだときの期待報酬」を表す「Q値」を更新する.
だんだん「どのマスでどちらに動けば良いか」がわかる.
その結果，最短経路を選べるようになる.

（他の方法との違い）
代表的なダイクストラ法や幅優先探索とは違う.
迷路の最短経路は，グラフ探索アルゴリズム（Dijkstra, BFS） で簡単に解ける.

（なぜ強化学習を使うのか？）
環境のルールがわからなくても学習できる.
探索アルゴリズムは地図や遷移モデルが与えられる必要がある.
強化学習は「行動してみて，結果と報酬を観察するだけ」で方策を学ぶ.
だから「未知の環境」「確率的に動く環境」でも適応可能と考えられる.
つまり，ゴールまでの道を試行錯誤で学ぶ　行動を学習で行なっている.

在庫管理問題をDP／強化学習で行う

ファイル名：　DP.ipynb　線形計画法（Dynamic Programming）
　　RL.ipynb　強化学習

目的：最適化問題を２つの方法で取り組み比較する.

そもそも，在庫管理問題とは？
・在庫がないと売るものがない
・しかし，在庫が多いと管理費用がかかる
・いっぽう，在庫を発注すると費用がかかる
おおよそ
・在庫が少なければ「たくさん発注する」方が良い
・在庫が多ければ「発注しない」方が良い
の方策になるが，ではどのくらい発注したらよいか最適な値があるか？

ー＞簡単な問題は最適化問題で解かれている.
DPを解くことで「その場で一番良い発注数」＝ optimal order a* が決まる.

方法：
１．DPで解いてみる：DP.ipynb
２．強化学習で解いてみる：RL.ipynb
３．DPと強化学習を比較する：RL.ipynb（の最後のほう）
４．強化学習の改良を試みる

ねらい：
・古典的には DP（動的計画法）で解ける問題
・でも RL を使うことで「最適解が求めにくい問題」にも応用できる
・強化学習は最適解を知らなくても，経験から近い解を学べることを理解！

ナップサック問題を強化学習で解く

ファイル名：　　Knapsack.ipynb

目的：組み合わせ最適化ができる問題をあえて強化学習で解いてみる.

ナップサック問題とは？
容量制限のあるカバンに価値の高いアイテムを詰めて，合計価値を最大化する問題.

方法：ナップサック問題を強化学習で解く考え方

状態 (state)：
　どのアイテムまで意思決定したか（インデックス i）
　残り容量 cap
行動 (action)：
　そのアイテムを 入れる (1)
　入れない (0)
　（ただし重さが残容量を超えるときは「入れる」は禁止）
報酬 (reward)：
　中間報酬は 0
　エピソード終了時に「選んだアイテムの価値合計」をまとめて報酬にする
方策 (policy)：
　ニューラルネットが「状態→確率分布（入れる／入れない）」を出力
　許されない行動はマスクして選ばせない
学習 (learning)：
　REINFORCE（方策勾配）で「価値が大きかった行動系列」の確率を高める
　移動平均ベースラインを使い，報酬のブレを減らして安定化

ねらい（くりかえし）：
・古典的には DP（動的計画法）で解ける問題
・でも RL を使うことで「最適解が求めにくい大規模ナップサック」にも応用できる
・強化学習は最適解を知らなくても，経験から近い解を学べることを理解！

フローショップスケジューリング問題を強化学習で解く

目的：フローショップスケジューリングを簡単な例から複雑な例へ拡張.

内容：

１．機械２台でJohnson規則で解く
　　機械２台ならば厳密解が存在する．それを確認.
　　利用ファイル：FlowShop_Johnson.ipynb

２．機械２台で強化学習で解いて，Johnson則と比較
　　強化学習で得た機械２台の結果が理想になるか？
　　利用ファイル：FlowShop_RL_2M.ipynb

３．機械３台で強化学習で解く（１）.
　　機械３台になると途端に難しくなる傾向がある．それを解く.
　　利用ファイル：FlowShop_RL_3M.ipynb

４．機械３台で強化学習で解く（２）.
　　ジョブはセット（S），処理（Process: P），取り外し（Remove: R）がある.
　　それらの時間を与えて，強化学習で解く.
　　利用ファイル：FlowShop_RL_3M_SPR.ipynb

５．機械n台で強化学習で解く（３）.
　　機械の台数を増やして解く.
　　利用ファイル：FlowShop_RL_nM_SPR.ipynb

ジョブショップスケジューリング問題を強化学習で解く

目的：ジョブショップスケジューリングを簡単な例から複雑な例へ拡張.

理解：ジョブとフローの違いの理解が必要です.

内容：

１．機械２台で強化学習で解いて，自分の結果と比較.
　　強化学習で得た機械２台の結果に勝てるか？
　　利用ファイル：JobShop_2M.ipynb

２．機械３台で強化学習で解く（１）.
　　ジョブはセット（S），処理（Process: P），取り外し（Remove: R）がある.
　　それらの時間を与えて，強化学習で解く.
　　利用ファイル：JobShop_3M_SPR.ipynb

３．機械３台で強化学習で解く（２）.
　　ガントチャートを見ると同時スタートあり．これは無理.
　　まずはセットにだけ人の制約を加えて解く.
　　利用ファイル：JobShop_3M_SPR2.ipynb

４．機械３台で強化学習で解く（３）.
　　機械によっては複数同時にセットできる場合がある．それも加味する.
　　ただしまずは人の制約は外す.
　　利用ファイル：JobShop_3M_SPR3_multi.ipynb

５．機械３台で強化学習で解く（４）.
　　４に人の条件をつける．ただしセットのときだけ.
　　利用ファイル：JobShop_3M_SPR3_multi2.ipynb

６．機械３台で強化学習で解く（４）.
　　５に人の条件をさらにつける．つまり取り外しも重ならないように制約.
　　利用ファイル：JobShop_3M_SPR3_multi3.ipynb

ジョブショップスケジューリング問題を強化学習で解く（２）

目的：
・機械３台，操作者１人．３台のうち１台は２つワークをセットできる条件で最適化.

利用ファイル：
JobShop_3M_SPR3_multi4.ipynb

データの説明：
これをリアルデータから読み取って設定できるようにする.

#全部で6つのジョブ（製品や作業のまとまり）があるという意味です.
N_JOBS = 6

#機械は3台あります．M0，M1，M2 と番号を振っています.
N_MACH = 3

#各ジョブは3つの工程（オペレーション）を順番に通る必要があります.
N_OPS = 3

#各ジョブがどの順番で機械を使うかを指定しています.
#例えば Job0: [0,1,2] は，Job0 は M0 → M1 → M2 の順に処理されるという意味です.
routes = np.array([
    [0, 1, 2], # Job0: M0 → M1 → M2
    [1, 2, 0], # Job1: M1 → M2 → M0
    [2, 0, 1], # Job2: M2 → M0 → M1
    [0, 2, 1], # Job3: M0 → M2 → M1
    [1, 0, 2], # Job4: M1 → M0 → M2
    [2, 1, 0], # Job5: M2 → M1 → M0
], dtype=np.int32)

#S1 と S2
#セット時間（準備時間）です．人の作業の時間です.
# S1 は 1本目のセット（長めの時間）.
# S2 は 2本目のセット（同じ機械で同時に2つ扱える場合，短めの時間）.
# ２台セットできない場合は適当な値でよいです.
S1 = np.array([[4,3,5],[5,4,3],[4,5,4],[3,4,4],[5,3,5],[4,4,3]], dtype=np.int32)
S2 = np.array([[2,2,3],[3,2,2],[2,3,2],[2,2,2],[3,2,3],[2,2,2]], dtype=np.int32)

#プロセス時間（加工時間）です.

#これは機械が動いている時間で，人は関与しません.
#仮の値として，セットやリムーブよりずっと長くしています.
P = np.array([[40,55,35],[35,45,60],[55,30,45],[30,65,40],[50,35,45],[45,50,30]], dtype=np.int32)
#リムーブ時間（取り外し時間）です.
#これは人の作業です．工程が終わった後に，製品を機械から取り外す時間です.
R = np.array([[13,12,13],[13,13,12],[12,13,13],[12,12,12],[13,12,12],[12,12,12]], dtype=np.int32)

#機械ごとに同時にセットできるスロット数です.
#M1だけ2つ扱えるように設定してあります.
SET_CAP_MAP = {0:1, 1:2, 2:1}

処理の流れ：

１．入力データの定義
上で説明した routes，S1，S2，P，R，SET_CAP を配列で与えます.

２．スケジューラの実行
・各ジョブの工程（Set → Process → Remove）を順番にシミュレーションします.
・オペレータ（人）は同時に1つしか作業できない.
・Set と Remove の時間は重なりません.
・前の工程が Remove まで完了しないと次の工程に進めません.

３．スケジュール表（イベント列）の生成
・各イベント（Set/Process/Remove）が「いつ開始して，いつ終了したか」を
　表に書き出します.
・CSVやJSONLの形で保存できます.

４．推定した出力
４．１　イベント表
列は phase, job, job_name, op, machine, machine_name, slot, start, end, dur, uses_operator です.
例えば
Set, Job0, Op0, M0, start=0, end=4, dur=4, uses_operator=True
は
「Job0の最初の工程をM0 にセットする作業Op0を,
　0時刻から4時刻までオペレータ（今回は一人だけしかいないので出てこない）が
　行う．かつ，これは人が行う作業である．」
という意味です.

４．２　ガントチャートと推定時間，単純なシナリオとの比較.
機械ごとの時間軸に，どのジョブがいつ処理されていたかを色で表示します．Set はハッチング，Process は塗りつぶし，Remove は別の色で区別されます．これで直感的に流れを理解できます．また，

Cmax (final): 454

として，強化学習では，４５４時刻によって処理されることが示されます.

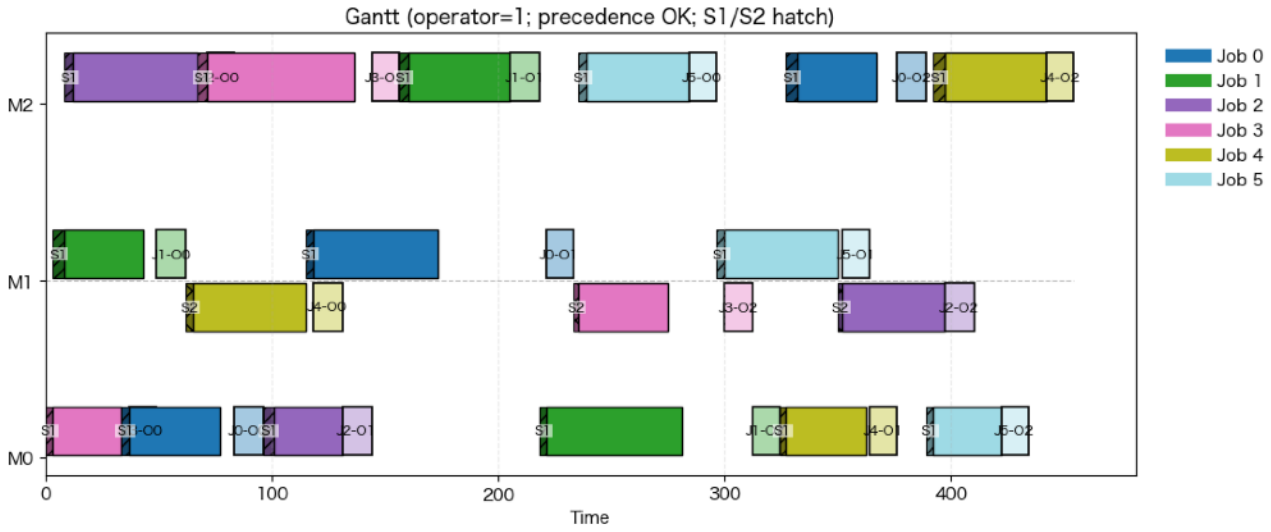

図１　ガントチャートの例．ジョブごとに色分け．セット，プロセス，リムーブで同じ色の濃度で表現（セット：濃い，プロセス：中，リムーブ：薄い）．機械M0とM2は１つしかワークを持てない．M1は２つワークを持てるので２段で表示．２つセットされた場合には，セットの領域をXでハッチング．１つしかない場合には／でハッチング.

単純なシナリオは，
「先に全ジョブの工程0（最初の工程）をまとめ，次に全ジョブの工程1，最後に工程2」です．素直に，Op0,Op1,Op2をこなしていく手順です．この場合には，

Cmax (final): 491

となり，４９１時刻（＞４５４）で処理されることがわかります.

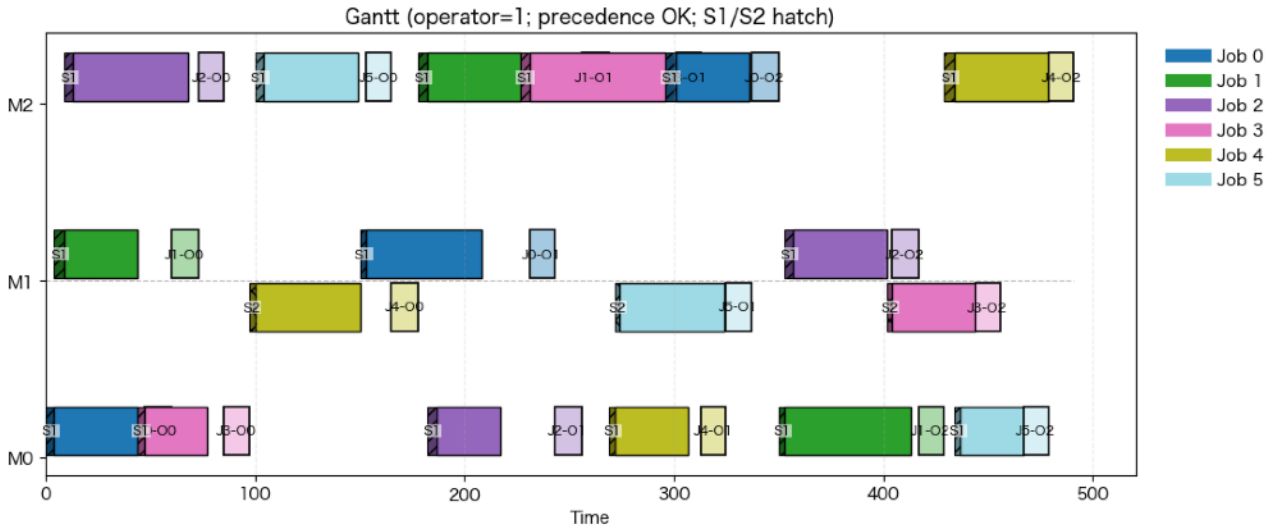

図２　単純シナリオのガントチャートの例.

要注意：本当に正しいか，矛盾がないかは，原は未確認です！Removeが可視化されていないように見えますが，薄い／時間が短いだけかも.

４．３　オペレータのタイムライン

オペレータがいつどの作業をしていたか（Set/Removeだけ）をまとめた表です．人の稼働率や待機時間が分析できます.

表１　タイムラインの例.

s	e	d	tasks
0	13	13	['Set Job0-O0 on M0', 'Set Job1-O0 on M1', 'Set Job2-O0 on M2']
44	104	60	['Set Job3-O0 on M0', 'Remove Job0-O0 on M0', 'Remove Job1-O0 on M1', 'Remove Job2-O0 on M2', 'Remove Job3-O0 on M0', 'Set Job4-O0 on M1', 'Set Job5-O0 on M2']
150	187	37	['Set Job0-O1 on M1', 'Remove Job5-O0 on M2', 'Remove Job4-O0 on M1', 'Set Job1-O1 on M2', 'Set Job2-O1 on M0']
227	274	47	['Set Job3-O1 on M2', 'Remove Job0-O1 on M1', 'Remove Job2-O1 on M0', 'Remove Job1-O1 on M2', 'Set Job4-O1 on M0', 'Set Job5-O1 on M1']
296	357	61	['Set Job0-O2 on M2', 'Remove Job3-O1 on M2', 'Remove Job4-O1 on M0', 'Remove Job5-O1 on M1', 'Remove Job0-O2 on M2', 'Set Job1-O2 on M0', 'Set Job2-O2 on M1']
402	437	35	['Set Job3-O2 on M1', 'Remove Job2-O2 on M1', 'Remove Job1-O2 on M0', 'Set Job4-O2 on M2', 'Set Job5-O2 on M0']
444	456	12	['Remove Job3-O2 on M1']
467	491	24	['Remove Job5-O2 on M0', 'Remove Job4-O2 on M2']

s: start, e: end, d: duration, start と endの時間区間にオペレータが作業していたことを表します.

dur：作業の合計時間（end – start）です.

tasks：その区間でオペレータが担当した作業のリストです.

（例）

[Set Job0-O0 on M0, Remove Job1-O0 on M1]
は,
「Job0 の最初の工程を M0 にセットした後，Job1 の最初の工程を M1 から取り外した」という流れ.

start=0, end=13, dur=13
tasks = [Set Job0-O0 on M0, Set Job1-O0 on M1, Set Job2-O0 on M2]
は,
・時刻0から13までの13単位時間，オペレータはずっと作業していた.
・そこでは，まず Job0 を M0 にセット，次に Job1 を M1 にセット.
　さらに Job2 を M2 にセットした.
・ここでは「順番に」こなしていることを意味し，同時に3つをやったわけではない.

４．４　まとめ
・オペレータのタイムラインは，人が関わった作業の履歴を時系列にまとめたもの.
・各区間（start〜end）は「オペレータが休まず作業を続けていた時間」.
・tasks のリストは，その間にこなした作業の順番を表す.

５．生成したイベントをLLMへ送って文書化してみる
（方法：実装した関数）
out_text = ask_gpt5_about_schedule("./schedule_events.csv", model="gpt-4o-mini", stream=True)

GPT4o-miniの出力例：
1) 時系列サマリ
- Job0のセットとプロセスが最初に行われ、終了後にRemoveされる。
- Job1、Job2、Job3が順次処理され、各セット後にプロセスが続く。
- 各Jobの終了後にはRemoveが実施され、次のジョブが開始される。
- Job4とJob5はそれぞれセット、プロセスを経てRemoveされる。
- 各ジョブの処理時間は異なるが、オペレータと機械は一貫して管理されている。

2) ボトルネック分析
- **機械M0**:
  - 時間帯 44〜77: M0はJob0とJob3を処理中（占有状態）。
  - 時間帯 183〜217: M0でJob2が処理中（占有状態）。
  - 時間帯 301: M0でRemoveが行われている。

- **機械M1**:
  - 時間帯 97〜150: M1はJob4を処理中。
  - 時間帯 182〜274: M1でJob1がセット/プロセス中（オペレータ待機）。

- **オペレータ**:
  - 時間帯 153〜165: Job0がRemove待ちでM1がオペレータ過負荷。
  - 時間帯 417〜429: Job1がRemoveされるまでオペレータ待機。

3) 改善アイデア
1. **ジョブ順序の最適化**: Removeを非同期で実施するなど、後続ジョブのセットアップを早め、設備の空き時間を最小化。
2. **段取りの効率化**: セット作業の時間短縮や事前準備を行い、オペレータの待機時間を削減。
3. **並列化の導入**: 複数の機械を使用して、同時進行で異なるJobのプロセスを進め、全体のリードタイムを短縮。

4) 重要な前提と矛盾の確認
- ジョブ順序に関しては、各Removeが完了してから次の工程（Set）が開始されており、矛盾は無い。
- 機械占有に関しても、Removeの施行が完了するまで他のJobが影響を受けていないため、一貫している。
- オペレータの役割についても、SetとRemoveで使われており、適切なタイミングでの使用が確認でき、矛盾は無い。

問題点：
・ワークをリムーブしたときに，別のワークを新たにセットできるのか？
・リムーブする前に新しいワークをセットできるのか？：多分できない.
・リムーブの時間がガントチャートで見にくい：表示方法の工夫が必要.
・セットは続けて行った方がよいはず？
・リムーブも続けて行った方がよいはず？
・機械間での人の移動時間は無視しているが，よいか？
　→どの機械の間での移動も同じ時間.

探索方針：
・悩んだら条件を単純化すること
・評価結果が正しいか，検証班（検証時期）を必ず設ける
・リアルデータから必要な項目を抽出して入力できるようにする