最適停止問題の方策と評価例

Hirotsugu Seike

2026 年 1 月 2 日

1　導入

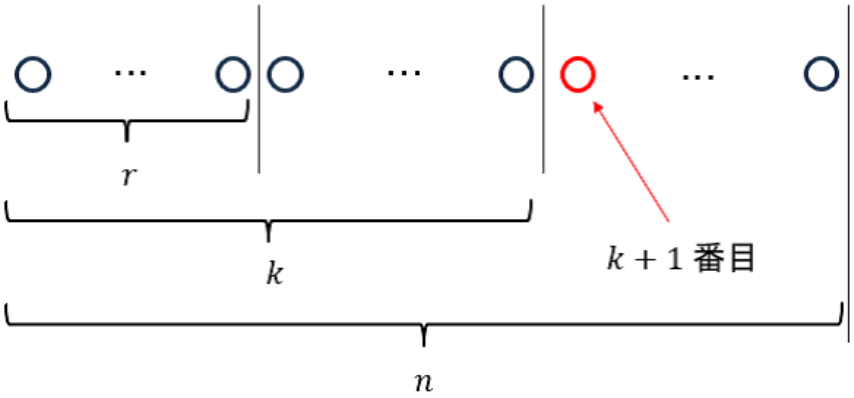

図 1: 最適停止問題の各種パラメータイメージ

最適停止問題 (optimal stopping) の一つに秘書問題 (secretary problem) が存在する. 秘書問題では, n 人の候補を面接し, 最も優秀な一人を採用することを目的とする. 候補者は, ランダムな順序で面接を受ける. 各面接において, 雇用者は採用・不採用を判定し, 採用となった時点で面接は終了する. 一度, 不採用となった候補者を, 後から採用することはできない. 候補者は, 全候補者を適切な順序尺度で評価することができる.

このような条件の下で, 雇用者は一番優秀な秘書を採用したいということである. ヒューリスティックには, ある程度の人数の面接を行った後に, それ以上に優秀な人が現れたら採用とするのが適当と考えられる (強化学習に明るい人は, 探索 (Exploration) と活用 (Exploitation) を行っていると考えてもよい). 図 1 にその様子を示す. つまり, r 人の候補者を面接し, その中で最も優秀と思われた人より優秀な人が現れたら採用するということである. この方策 (Policy) に基づいて, 最も優秀な人が採用される確率が高くなるような r を求める.

$k+1(k \geq r)$ 人目が最も優秀な人である確率は $1/n$ である. また, k 人目までに別の人が採用されない確率は r/k である (k 番目までで一番順位が高い人が前半の r 人に含まれる必要がある). 以上より, 最も優秀な人が採用される確率 P は次式を満たす (一部, 近似を仮定している点には注意する).

$$P = \sum_{k=r}^{n-1} \frac{1}{n} \cdot \frac{r}{k} = \frac{r}{n}\left(\frac{1}{r} + ... + \frac{1}{n-1}\right), \tag{1}$$

$$\sim \frac{r}{n}\left(\log(n) - \log(r)\right), \tag{2}$$

$$= -\frac{r}{n}\log(\frac{r}{n}) \leq \frac{1}{e}. \tag{3}$$

$x\log(x)$ が凸関数であるため, $-x\log(x)$ は微分が 0 となる点で最大となる. つまり, $r = n/e \sim 0.37n$ で, 方策を探索から活用にシフトすると, 最も優秀な人を採用できる確率が最大となる.

2　シミュレーションによる方策の違いによる利得比較

確率を利得 (Advantage) と考えて, シミュレーションした結果が図 2 である. 図 3 にそのソースコードを示す.

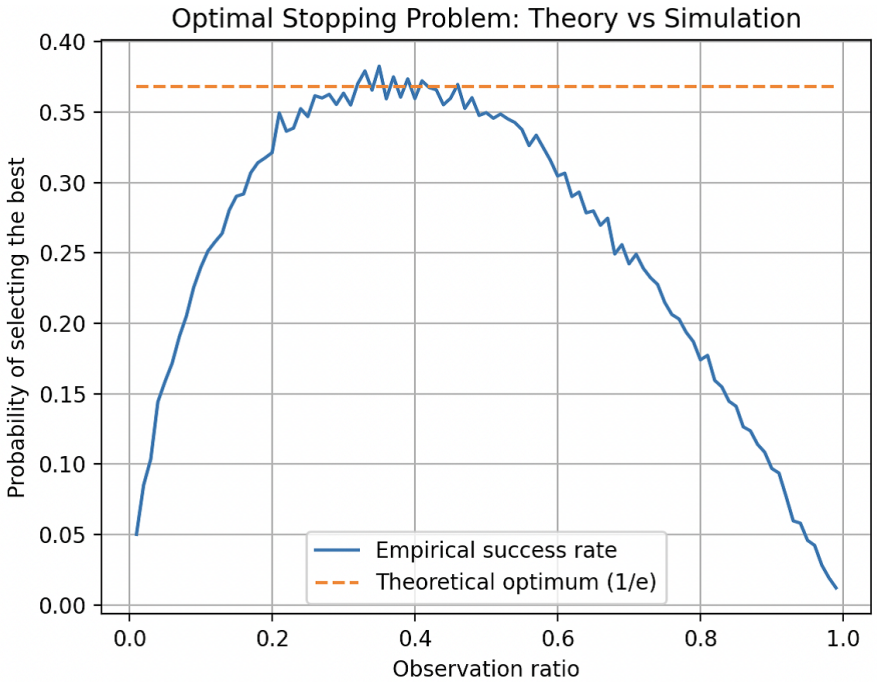

図 2: 最適停止問題の各種パラメータイメージ

```
import random
import math
import matplotlib.pyplot as plt


def simulate_secretary_problem(
    n: int,
    observation_ratio: float,
    trials: int = 5000
) -> float:
    """
␣␣␣␣秘書問題のシミュレーション
```

```
␣␣␣␣-␣n:␣候補者数
␣␣␣␣-␣observation_ratio:␣最初に観察だけする割合
␣␣␣␣-␣trials:␣試行回数
␣␣␣␣戻り値:␣最良候補を選べた確率
␣␣␣␣"""
    success = 0
    threshold = int(n * observation_ratio)

    for _ in range(trials):
        # 真の順位（n が最良）
        candidates = list(range(1, n + 1))
        random.shuffle(candidates)

        # 観察フェーズ
        best_so_far = max(candidates[:threshold])

        # 選択フェーズ
        chosen = None
        for c in candidates[threshold:]:
            if c > best_so_far:
                chosen = c
                break

        # 誰も選ばなかった場合は最後の人
        if chosen is None:
            chosen = candidates[-1]

        # 本当に最良（）を選べたかn
        if chosen == n:
            success += 1

    return success / trials

def main():
    n = 100                  # 候補者数
    trials = 5000            # 各点の試行回数
    ratios = [i / 100 for i in range(1, 100)]

    empirical = [
        simulate_secretary_problem(n, r, trials)
        for r in ratios
    ]

    theoretical = [1 / math.e for _ in ratios]

    # --- plot ---
    plt.figure()
    plt.plot(ratios, empirical, label="Empirical␣success␣rate")
    plt.plot(
        ratios,
        theoretical,
        linestyle="--",
        label="Theoretical␣optimum␣(1/e)"
    )
    plt.xlabel("Observation␣ratio")
    plt.ylabel("Probability␣of␣selecting␣the␣best")
    plt.title("Optimal␣Stopping␣Problem:␣Theory␣vs␣Simulation")
    plt.legend()
```

```
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    main()
```

図 3: サンプルコード

参考文献

付録 A　特になし