



# スタートアップゼミ Assignment2

2026/4/20

B4 鷺野 萌花



# 発表内容

## 1. テキストのまとめ

1.1 最適課金の定式化、第一最善 (2.5節、2.8節)

1.2 Day-to-Dayダイナミクスと交通配分 (2.7節)

- ・ (余裕があれば) 動的交通配分

1.3 テキストの内容を踏まえた問1、問2の確認

## 2. 個人研究

2.1 第二ベスト課金の実装・厚生分析と考察

### < Appendix >

- ・ LLM利用について

# 1.1 最適課金の定式化、第一最善（2.5節、2.8節）

## ●混雑の外部性

個人の合理的行動の集積が必ずしも社会全体にとって望ましい結果をもたらすとは限らない（例: Braessのパラドックス）。

## ●Wardropの第2原則

ネットワーク全体の総旅行時間が最小となるように交通量が配分される→この原則を実現する配分がシステム最適配分（SO）

SO-Primal

$$\min_{\mathbf{x}, \mathbf{f}} Z_{SO}(\mathbf{x}) = \sum_{a \in A} x_a t_a(x_a) \quad \leftarrow \text{UEとの違い: 総旅行時間の最小化} \quad (2.49)$$
$$\text{s.t. } x_a = \sum_{rs \in \Omega} \sum_{k \in K_{rs}} f_k^* \delta_{a,k}^* \quad \forall a \in A$$
$$\sum_{k \in K_{rs}} f_k^* = q_{rs} \quad \forall rs \in \Omega$$
$$f_k^* \geq 0 \quad \forall k \in K_{rs}, \forall rs \in \Omega$$

# これを解くとどうなる？

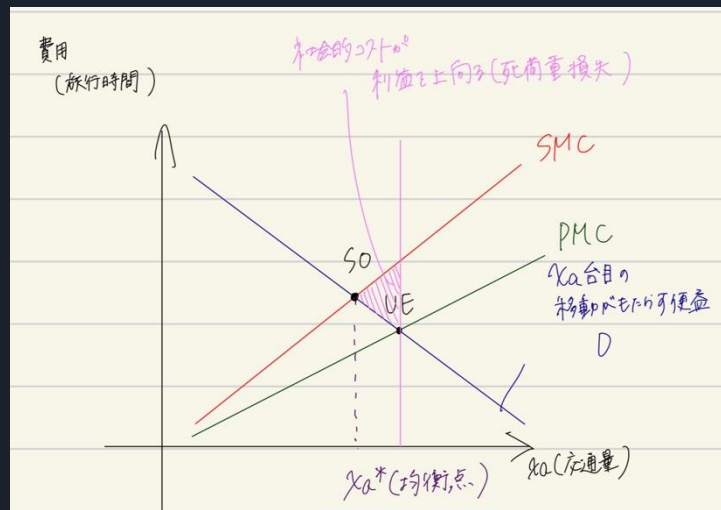
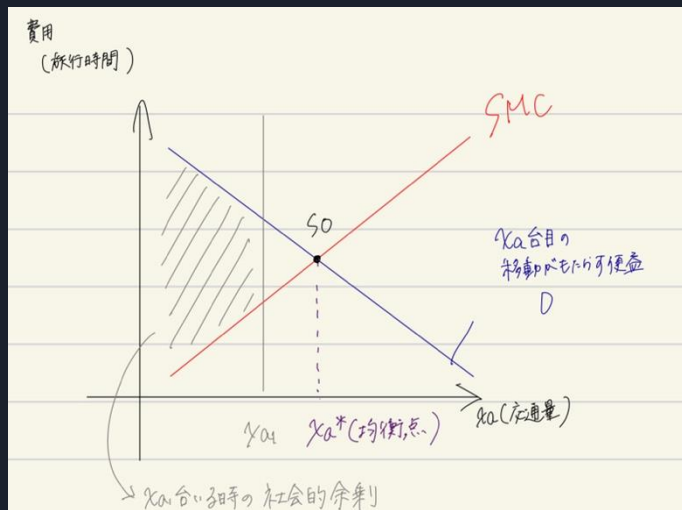
- ・ UEの時と同じように、Frank-Wolfe法を用いることができる（目的関数が凸であるため）
- ・ しかしここで、最短経路探索のコストとして用いるのは私的限界費用ではなく**社会的限界費用**である（自分がその時道路に入ることで生じるコストだけではなく、自分の影響で他人にかかっているコスト増分（=外部性）まで考慮するということ）

$$\frac{\partial Z_{UE}}{\partial x_a} = t_a(x_a)$$

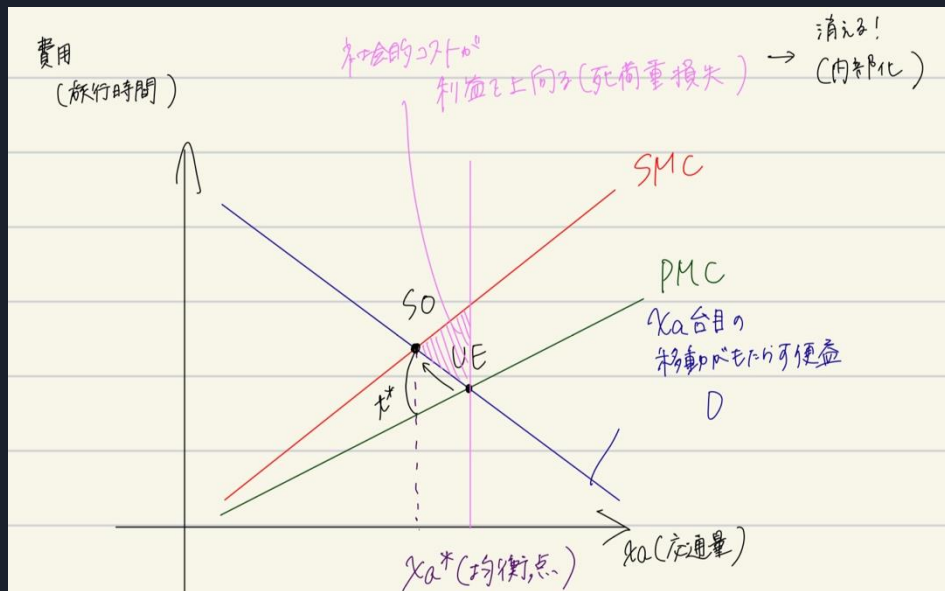
$$\frac{\partial Z_{SO}}{\partial x_a} = \frac{d}{dx_a} [x_a t_a(x_a)] = t_a(x_a) + x_a t'_a(x_a)$$

↑ 他者に与えている遅延

# 経済学的にどういう意味合いなのか



# 経済学的にどういう意味合いなのか



- 総旅行時間を最小化するSO配分は、適切な**混雑課金**によって達成でき、外部費用は内部化できる
- 混雑課金は、経済学のピグー税の考え方で説明できる
- 課金学を再配分することでパレート最適になる

# 実際、どのように課金を行うのか（2.8節）

・ネットワーク上の全てのリンクに、任意の大きさの課金を設定できるという条件下での最適課金を**第一最善課金（First Best）**という

→この場合、各リンクに先ほどと同じように課金すれば、社会的余剰が最大化され、政府は得られたお金を再配分することで、パレート改善が実現する

・しかし、**全てのリンクに課金するのは無理**

→**Second Best**な課金を行う

この場合課金されたところを避けようとするので、

**バイレベル計画問題**を解く必要がある：

政府などによる料金体系の設定→その課金におけるUE配分を求める→


それに基づきより良い料金体系を設定→・・・



# Day-to-Dayダイナミクスと交通配分（2.7節）

- ・ 今までの交通量配分における「均衡」の議論は、「均衡状態がどこか」についてで、「**どのように収束するか**」については議論がされていなかった
- ・ Day-to-dayダイナミクスでは、完全情報ではなくユーザーが有限合理性に基づいて日々行動を変えていくことを前提とする（昨日はこの道が混んでいたから今日はこちらの道を通ろう、というような意識）
- ・ **進化ゲーム理論**によって説明できる





# 進化ゲーム理論

●ポテンシャルゲーム：全てのプレイヤーの利得の変化が一つのグローバルな関数の変化に等しいゲーム

・交通配分もポテンシャルゲーム

・ポテンシャル関数としてBeckmann変換の目的関数を採用すると、**一人のドライバーが経路kから経路lに変更した時のZの変化量は、そのドライバーの旅行時間の変化に等しい**

→どうということか？

→そこにいるドライバーが自分の都合で最短経路に動けば、Zを減らす方向に動く

→いつか、誰も経路変更で得をしない状態が発生するが、それはZが最も小さい時であると解釈できる

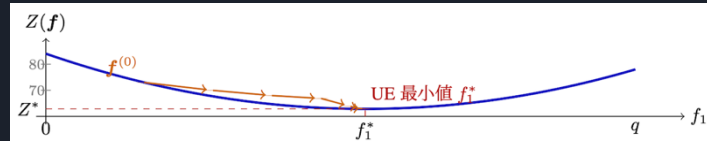
# Day-to-dayダイナミクスにおける改訂プロトコル

- ・ Day-to-dayダイナミクスは進化ゲーム理論で説明可能
- ・ Nash均衡分析ではプレーヤーは常に最良の戦略を選択することが前提だが、Day-to-dayモデルでは利用者は確率的、漸次的に戦略を変更できる→乗り換えるかどうかを社会状態に基づいた確率などで表せる（改訂プロトコル）
- ・ 改訂プロトコルに基けば流出入がわかるので、進化動学として定式化して表せる
- ・ どんなふうに戦略を変えるかについては、いろいろなモデルがある

Smithダイナミクス	$\dot{x}_i = \sum_j x_j [F_i - F_j]_+ - x_i \sum_j [F_j - F_i]_+$	コスト差に比例して乗り換え
レプリケーター動学	$\dot{x}_i = x_i (F_i - \bar{F})$	平均と比較
最良応答動学	$\dot{\mathbf{x}} \in \text{BR}(\mathbf{x}) - \mathbf{x}$	一気に便利な方に乗り換え

# Day-to-Dayダイナミクスで本当に収束するの？

→する（進化動学に幾つかの条件を加えることで、**時間とともに必ず減るポテンシャル関数**（Lyapunov関数）を作ることができるため！）



→何が凄いのか？

**正相関条件（コストが高い経路を避ける傾向）を持つ日々の経路変更行動が続けば、長期的にはUEが実現する**

→この前提があるからこそ、Day-to-dayダイナミクスがあれば、何か政策を変えてみた時に、均衡状態に落ち着くまでの過程を確認できる

（例えば、適切に道路に課金した状態でこのダイナミクスを実装すれば均衡に落ち着いた時はSOに近い状態になるが、それまでの過程を確認することができる）

# 課題

- ・ 山手線の（大体）内側の幹線道路における交通量配分に関する問題
- ・ 課金した場合と、課金しなかった場合での総旅行時間の比較

## Section 2：問1 ベースライン UE 配分（課金なし）

day-to-day ダイナミクスシミュレーションを用いて、課金がない状態での\*\*利用者均衡（User Equilibrium; UE）\*\*を求め、均衡総旅行時間を

$$T_{UE} = \sum_a t_a(x_a) x_a$$

を計算する。

各リンク  $a$  の旅行時間は **Bureau of Public Roads (BPR)** 関数

$$t_a(x_a) = t_{a0} \left( 1 + 0.15 \left( \frac{x_a}{C_a} \right)^4 \right)$$

で与えられる ( $t_{a0}$ : 自由流旅行時間 [h],  $C_a$ : リンク容量 [台/h])。

復習：Day-to-dayダイナミクスではポテンシャル関数が収束し、長期均衡は静学的UEと一致する

# Day-to-dayダイナミクス上で用いられている動学

```
# 初期解: 自由流 AON
x = all_or_nothing(G, ei, od, ei.t0.copy(), n_jobs=n_jobs)

t_history: list[float] = []
x_history: list[np.ndarray] = []

for t in range(max_iter):
    # 課金の有効・無効を切り替え (過渡期分析用)
    active_toll = (
        np.zeros(ei.n_edges)
        if (toll_start_iter is not None and t < toll_start_iter)
        else toll_arr
    )

    costs = ei.travel_times(x, active_toll)
    x_aon = all_or_nothing(G, ei, od, costs, n_jobs=n_jobs)

    x_new = (1.0 - step) * x + step * x_aon

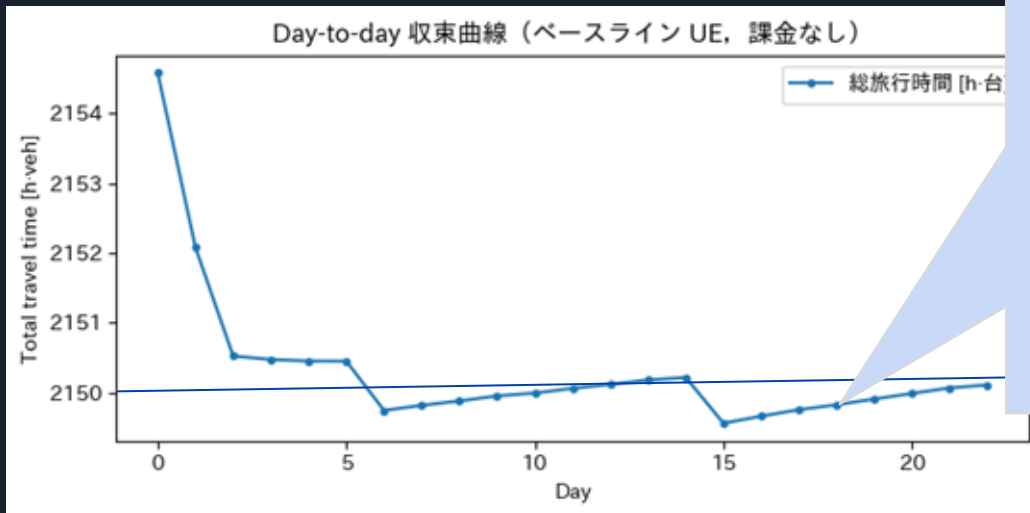
    t_history.append(ei.total_travel_time(x_new))
    if x_save_interval > 0 and (t % x_save_interval == 0):
        x_history.append(x_new.copy())
```

$$x_{t+1} = (1 - \eta)x_t + \eta x_{aon}$$

定めた課金額のもとでAON配分を行うが、全ての車が一気に最短経路に向かうのではなく、実際には一部だけ（今回であれば10%）がそちらに動く

# 問題1：課金なしVer (toll\_arr=None)

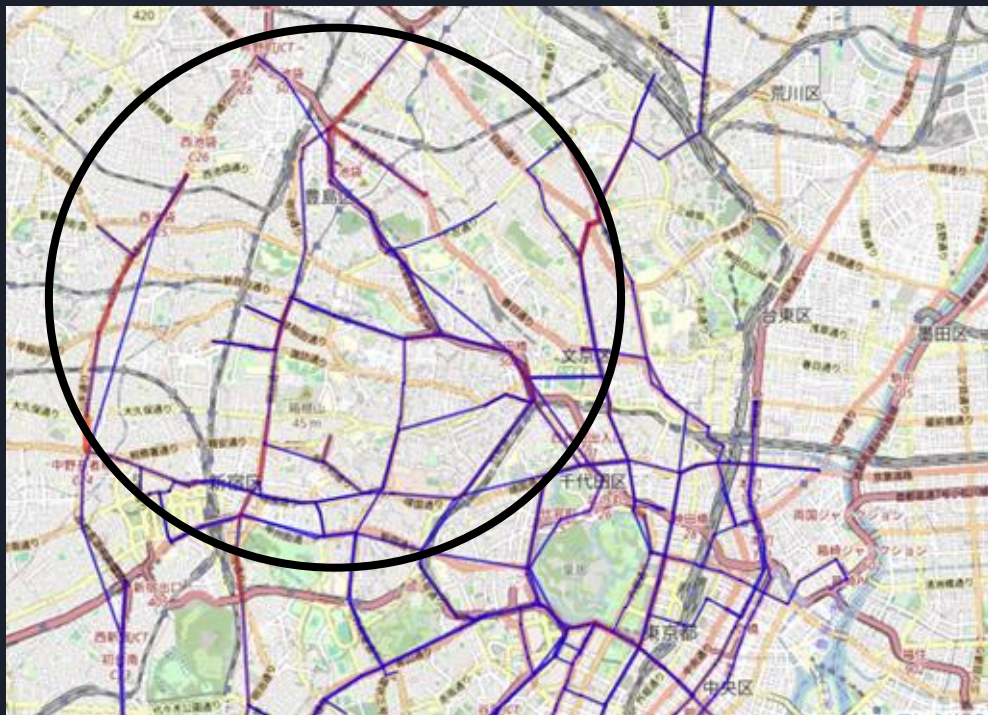
- ・ 収束日数：23日
- ・ 総旅行時間は2150.1078 (台・時間)
- ・ 収束に至るまでの過程は以下



なぜ振動している？  
→ 動学の一部にAONが用いられている (不連続)  
例えば2経路で考えるとしたら、A経路に車が集まっていたのが、コストのわずかな差でB経路が優勢になると、AONがB経路の方に配分するので、均衡ラインを急に飛び越える

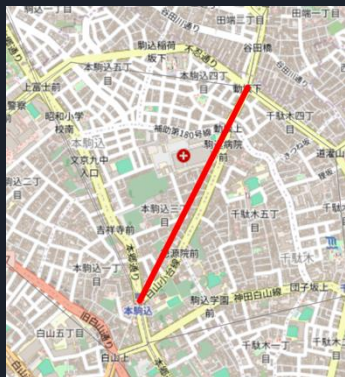
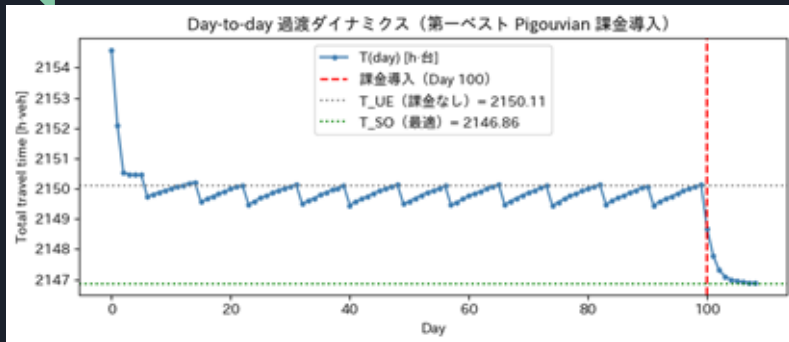
このノコギリのような形は、 $\eta$ を変えれば変わる

# 均衡時の混雑率 $x/C$ が高いリンク



赤色が混雑率の高いリンク

## 問2：課金あり ver (First-best：すべての道路に課金)



・課金なしの状態では総旅行時間の最小化をFrank-Wolfe法を使って行った場合/それによって導くことができる最適な課金をしてDay-to-dayダイナミクスを考えた時 の総旅行時間の比較

・最大課金は0.16 (分ぶん)：ここに時間価値をかければ具体的な金額がわかる

・最小化を行った場合は2146.8598 (台・時間)

・100日目から課金を開始→コストが急に変化し、2146.8812 (台・時間) 付近で収束 (最小化した時とほぼ同じになる)

・課金しないときは2150.1078 (台・時間) だったので改善はされている

・最も課金されたリンクは本駒込のあたり (リンク容量に依存)

## 課題A：第二ベスト課金



課金可能なリンク

```
toll_fb = pigouvian_toll(ei, x_so, scale=1.0)
toll_sb = toll_fb * ei.tollable
```

- ・ 課金可能なリンク（高速道路など）に対してだけ、問2で求めたそのリンクのPigouvian課金を施す
- ・ 最大課金は0.09分ぶん、収束日数は100日後に課金をした後13日、総旅行時間はUEの際の**2150.1078**（台・時間）から**2150.0788**（台・時間）と、**UEの時にほぼ逆戻り**してしまった
- ・ つまりFirst-Bestな課金を課金可能なリンクにしたところであまり意味はない

# First-BestとSecond-Bestの比較

```
# ===== FB vs SB 比較 =====

mask_toll = ei.tollable
mask_non  = ~ei.tollable

# 課金リンク
flow_toll_fb = x_fb[mask_toll].sum()
flow_toll_sb = x_sb[mask_toll].sum()

# 非課金リンク
flow_non_fb = x_fb[mask_non].sum()
flow_non_sb = x_sb[mask_non].sum()

print("==== FB vs SB 比較 ====")

print("\n[課金リンク]")
print("FB:", flow_toll_fb)
print("SB:", flow_toll_sb)
print("変化:", flow_toll_sb - flow_toll_fb)

print("\n[非課金リンク]")
print("FB:", flow_non_fb)
print("SB:", flow_non_sb)
print("変化:", flow_non_sb - flow_non_fb)
```

==== FB vs SB 比較 ===

[課金リンク]

FB: 196606.15735003847

SB: 195130.63729254674

変化: -1475.5200574917253

[非課金リンク]

FB: 319318.4052041303

SB: 324108.54584355786

変化: 4790.140639427584

- リンク交通量の合計を、課金リンクと非課金リンクについて、FBとSBで比較
- 課金したリンクでは交通量が減り、非課金のリンクでは増える（スピルオーバー）
- これは真のSecond-bestではないため、実際にはバイレベル計画問題を解く

# FBとSBの比較（社会的余剰=旅行時間の増分による損 + 課金による社会的な利益の増加）

```
# ===== 社会的余剰の変化(FB → SB)=====
# 1. 総旅行時間
T_fb = ei.total_travel_time(x_fb)
T_sb = ei.total_travel_time(x_sb)

dT = T_sb - T_fb # マイナスなら改善

# 2. 課金収入(時間単位)
revenue_sb = (toll_sb * x_sb).sum()

# 3. 社会的余剰(時間単位)
dW = -dT + revenue_sb

print("=== 社会的余剰の変化 (FB → SB) ===")
print(f"ΔT (総旅行時間) : {dT:.4f} [h・台]")
print(f"課金収入: {revenue_sb:.4f} [h・台]")
print(f"ΔW (社会的余剰) : {dW:.4f} [h・台]")

# 判定
if dW > 0:
    print("→ 社会的余剰は改善 (政策は効率的) ")
else:
    print("→ 社会的余剰は悪化 (非効率) ")
```

ΔT（総旅行時間）：3.1975 [h・台]  
課金収入：7.6696 [h・台] ΔW（社会的余剰）：  
4.4720 [h・台] → 社会的余剰は改善（政策は効率的）

社会的にプラスではある。

再配分する課金収入について考えると、大阪府の資料によれば人間の時間価値は38円/分くらいであるそうなので、ネットワーク全体の収入は1時間で17500円くらい？

今回は総トリップ数が23100台に設定されている（少ない）ためそうなる

# LLMの利用方法について

2.95式を説明してください

2.95式は、Vickreyポトルネックモデルにおいて\*\*「遅刻側（希望到着時刻  $t^*$  より後に到着する人）」の一般化費用\*\*を表しています。

このモデルでは、各通勤者の費用は

- 待ち行列による待ち時間費用
- 早着ペナルティ
- 遅着ペナルティ

の和で決まります。記号の意味は：

- $\alpha$  : 待ち時間1時間あたりの費用
- $\beta$  : 早着1時間あたりのペナルティ
- $\gamma$  : 遅着1時間あたりのペナルティ
- $w(t)$  : 時刻  $t$  に出発した人の待ち時間
- $t + w(t)$  : 実際の到着時刻

## 2.95式の形


遅刻する場合（到着が  $t^*$  より後）には：

$$C(t) = \alpha w(t) + \gamma(t + w(t) - t^*) \quad (2.95)$$

ありがとうございます。Second-bestの実装に移るのでどこを変えたらいいか教えてください。



思考時間: 2 ~ 3秒

はい。変える場所は「課金ベクトルを作っているところ」だけでほぼ足ります。



`assignment.py` の配分エンジン本体ではなく、`notebook` の `toll_fb = ...` を作っている部分の直すのが中心です。 

### まず結論

今たぶんこうなっていますよね。

```
Python   実行する  
toll_fb = pigouvian_toll(ei, x_so, scale=1.0)
```

これを second-best 用にすれば、次を追加します。

```
Python   実行する  
toll_fb = pigouvian_toll(ei, x_so, scale=1.0)  
toll_sb = toll_fb + ei.tollable
```

- 一般的な使い方
- ChatGPTのプロジェクトモードを使い、配布されたテキストをLLMに読ませて、数式のわからないところを説明してもらった。納得がいくまで何回も壁打ちできるのでとても助かる。ただし、グラフなど図形的な質問に答えるのは苦手。
- Second-bestの課金以降のところのコーディングはほぼやってもらった。今回は簡単なコードなのでどういう式で何が出力されるのかはわかるが、より複雑になるとそうもいかないので、LLMが出したコードはちゃんと見返すべき