

AIを用いた交通研究の事例紹介と考察

瀬尾亨（東京科学大学）

「ワークショップ：AIと交通研究」

第7回東京大学トランスポートハブリサーチセミナー

東京大学 2026-03

- Large Language Model (LLM) をはじめとするAIの発展が著しい
 - 本発表では、AIとは「LLMが核として組み込まれた自動システム」という意味で使う
- 我々の交通研究分野でも、AIをどうやればうまく使えるか、どう使うべきかを考える必要がある
- 交通研究の特徴
 - 広い時空間における人間行動×物理法則
 - 相反する価値判断や、囚人のジレンマ構造とその解決
- この点について、AIを用いて最近当研究室で行った研究を紹介し、考察・議論したい
- 本発表の位置づけ
 - 瀬尾はAIを使った交通研究に優れているというわけではない。AIがもっと得意な学生も多いはず
 - 「ちょっとやってみたので、ゼミ発表と一緒に勉強させてください」という発表
 - 当たり前の内容だらけだったらご容赦&ご指導ください
 - ワークショップの趣旨を踏まえ、議論のためにあえて極論を書いている箇所があります
 - 1年後にはまた状況が変わっている可能性は高い

DRL信号制御とドライバー経路選択における囚人のジレンマの成立条件

深層強化学習による適応的信号制御とドライバー経路選択行動の2×2ゲーム分析

背景

交通ネットワークにおける2つの意思決定主体

- **信号制御者**: システム全体の効率最大化を目指す
- **ドライバー**: 個人の旅行時間最小化を目指す (Wardrop第一原則 → UE)

既知の事実

- DRLによる適応的信号制御は定周期信号を上回る (Wei+, 2018; Zheng+, 2019)
- User Equilibrium (UE) は個人合理的だが社会最適 (SO) と乖離 → Price of Anarchy

見落とされている問題

- 多くのDRL研究はドライバーの経路選択を固定して評価
- DRL導入には**実装コスト** (センサ・計算資源・ML人材) が伴う

研究のギャップ: DRL導入コストを考慮した信号制御者 vs ドライバーのゲーム構造は未解明

研究目的

信号制御戦略とドライバー経路選択行動の 2×2 ゲームを定式化し、**DRL導入コスト c をパラメータとして囚人のジレンマ (PD) の成立条件を導出する**

	ドライバー SO (協調)	ドライバー UE (裏切り)
信号 DRL (協調)	CC: 相互協調	CD: 信号が裏切られる
信号 Fixed (裏切り)	DC: フリーライド	DD: 相互裏切り

- **ドライバー:** UEはNash均衡 → 裏切り (UE) が支配戦略 (Wardropの第一原則)
- **信号制御者:** DRL導入には追加コスト c が発生 → c が大きいとFixed (裏切り) を選好

ゲーム理論的定式化

信号制御者の一般化費用:

$$C_S(\sigma, \delta) = T_{\sigma}\delta + c \cdot \mathbf{1}[\sigma = \text{DRL}]$$

信号制御者の費用マトリクス:

	ドライバー SO	ドライバー UE
DRL	$T_{CC} + c$	$T_{CD} + c$
Fixed	T_{DC}	T_{DD}

Fixed（裏切り）が支配戦略となる条件:

- SOドライバーに対して: $c > T_{DC} - T_{CC} \equiv \Delta_{\text{coop}}$
- UEドライバーに対して: $c > T_{DD} - T_{CD} \equiv \Delta_{\text{selfish}}$
- したがって一般には: $c > \max(\Delta_{\text{coop}}, \Delta_{\text{selfish}})$

CCがDDをパレート支配: $c < T_{DD} - T_{CC}$

$$\max(T_{DC} - T_{CC}, T_{DD} - T_{CD}) < c < T_{DD} - T_{CC} \quad \text{のとき囚人のジレンマが成立}$$

(今回の実験では $\Delta_{\text{selfish}} > \Delta_{\text{coop}}$ のため, 下限は $T_{DD} - T_{CD}$ に簡約)

PD領域の幅 (今回の実験の簡約形) = $T_{CD} - T_{CC}$ (DRL制御下でのドライバー逸脱コスト)

シミュレータ: UXsim

UXsim (Seo, 2023) はPythonネイティブのメソスコピック交通流シミュレータ

交通流モデル: Kinematic Wave理論に基づくリンクモデル + ノードモデル

- リンク内の交通流動態（渋滞波の伝播・待ち行列形成）を物理的に再現
- 個々の車両（プラトーン）を追跡可能 → 車両単位の制御・経路指定が可能
- Pythonコーディングによるシミュレーション → 柔軟に操作可能

本研究で活用した主要機能:

UXsimの機能	本研究での用途	対応するゲーム要素
Dynamic User Optimum (DUO)	ドライバーの利己的経路選択	UE（裏切り）
<code>Vehicle.enforce_route()</code>	車両単位のSO経路配分	SO（協調）
信号制御API (<code>signal_phase</code>)	DRLによるリアルタイム制御	DRL（協調）
<code>signal=[30,30]</code> 定周期設定	定周期信号のベースライン	Fixed（裏切り）
<code>exec_simulation(duration_t2)</code>	10秒刻みのstep実行	RL環境のstep関数

UXsimの優位性: DUO経路選択と車両単位の経路固定を同一シミュレータ内でPython操作で切り替え可能
→ UE vs SO の公平な比較を単一フレームワーク上で実現

シミュレーション手法: 中央集権型DQN

共有エンコーダ + 交差点別行動ヘッドによる中央集権型アーキテクチャ:

$$Q_i(s, a_i) = h_i(f_\theta(s)), \quad i = 1, \dots, 5$$

状態: $s \in \mathbb{R}^{55}$ (5交差点 × 11次元: 車両数, 待ち行列, フェーズ, 経過時間)

報酬: $r_t = -\frac{1}{\alpha} \sum_{l \in \mathcal{L}} q_l(t)$ (全リンク待ち行列の総和, $\alpha = 50$)

Double DQN:

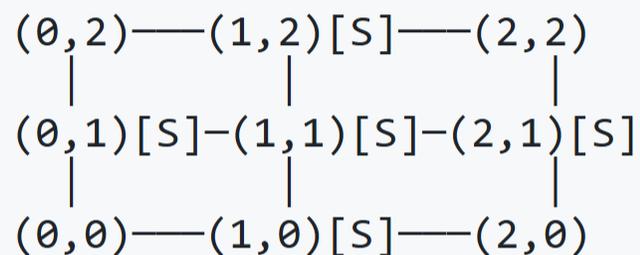
$$y = r + \gamma Q_{\theta'}\left(s', \arg \max_{a'} Q_\theta(s', a')\right), \quad \mathcal{L} = \frac{1}{N} \sum_i \text{SmoothL1}(Q_i - y_i)$$

フェーズ制約: $20s \leq \tau_{\text{green}} \leq 60s$

エンコーダ: 55 → 128 → 128 (ReLU) → ヘッド: 128 → 2 × 5交差点 | ϵ : 0.3 → 0.05 | $\gamma = 0.95$ | 300 ep訓練

実験設定

3×3グリッドネットワーク (9ノード, 24リンク, 5信号交差点)



リンク長 $L = 500$ m
自由流速度 $v = 15$ m/s
渋滞密度 $\kappa = 0.2$ veh/m
1車線
[S] = 信号交差点, 隅 = OD

需要: 対角4ペア (0.30 veh/s) + 辺4ペア (0.20 veh/s) × 2400 s = **4,800台**

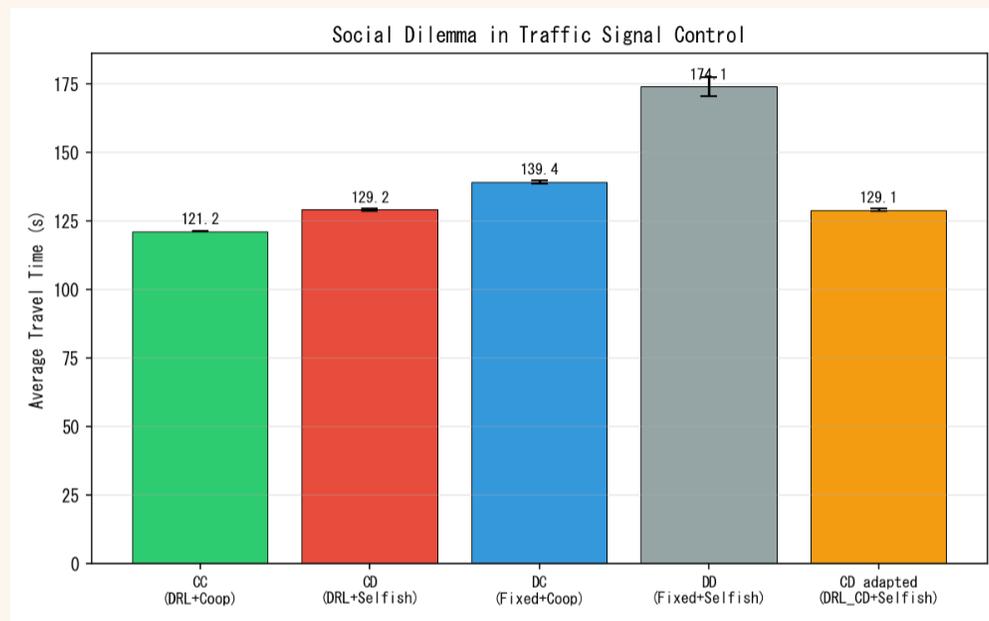
ドライバー行動:

- SO (協調) : 全最短経路 \mathcal{P}_{od} にラウンドロビン均等分散, `enforce_route` で固定
- UE (裏切り) : $t_{DUO} = 30$ s ごとに個人最適経路へ再ルーティング

信号制御: DRL = 中央集権型Double DQN / Fixed = [30, 30] s 定周期 (60 sサイクル)

評価: 各条件 **20シード** (seed 1000–1019), Welchの t 検定 ($\alpha = 0.05$)

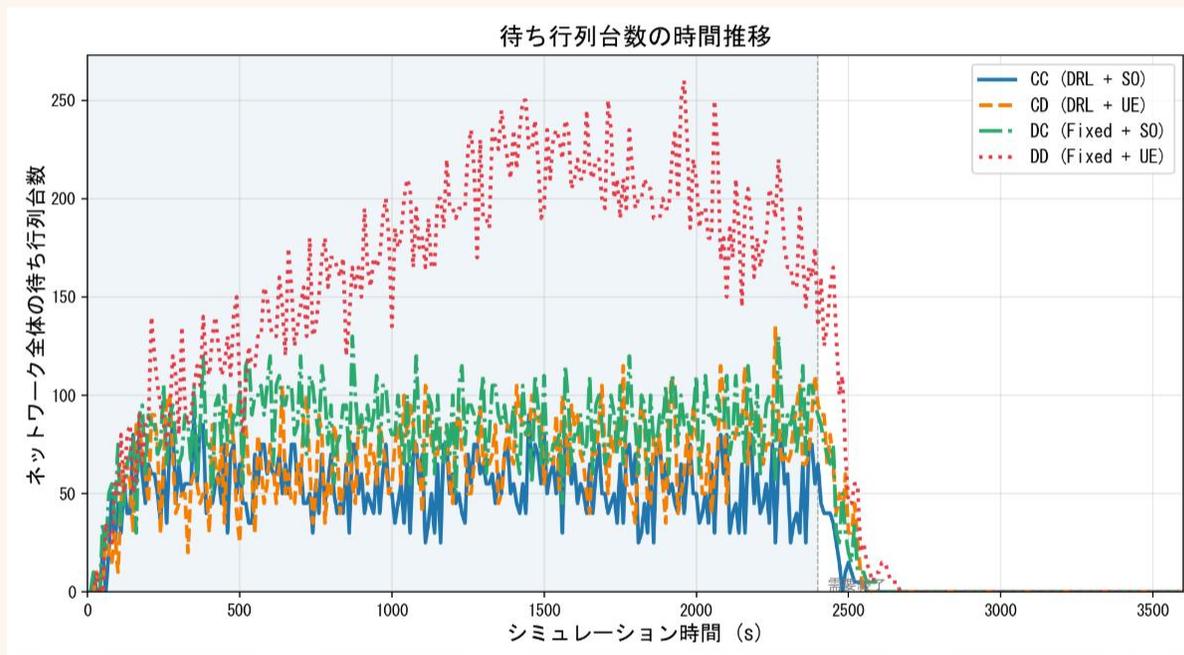
結果: 4条件のATT



条件	ATT平均 (s)	標準偏差	95% CI
CC (DRL + SO)	121.2	0.4	± 0.18
CD (DRL + UE)	129.2	1.2	± 0.54
DC (Fixed + SO)	139.4	1.4	± 0.63
DD (Fixed + UE)	174.1	7.8	± 3.43

待ち行列台数の時間推移

4条件のネットワーク全体の待ち行列台数（10秒刻み）：



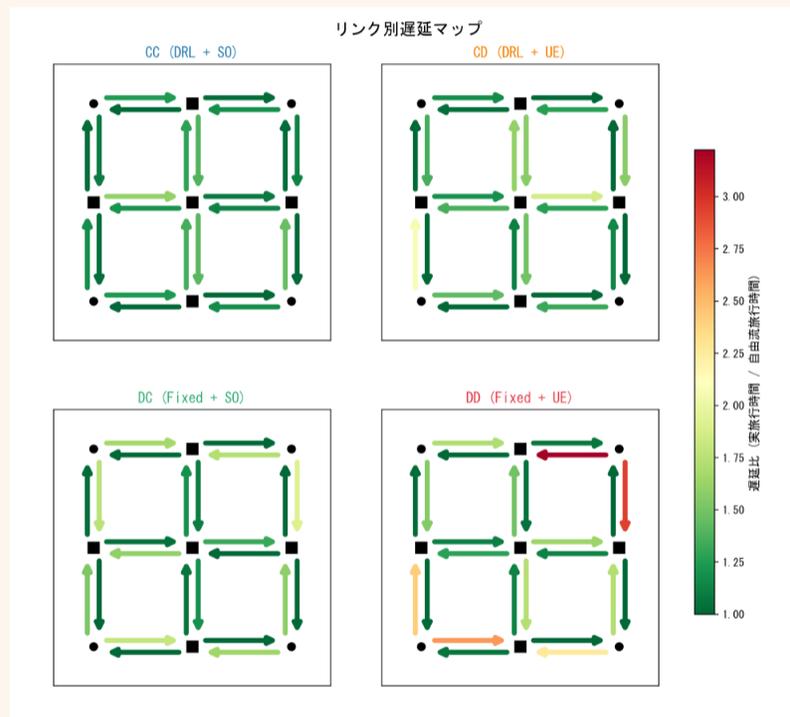
CC (DRL+SO): 待ち行列が常に低水準 → DRLが渋滞蓄積を未然に防止

DD (Fixed+UE): 需要期間中に待ち行列が大量蓄積（ピーク～250台）し、需要終了後も解消に時間を要する

信号制御（DRL vs Fixed）が待ち行列水準に最も大きく影響

リンク別遅延マップ

各リンクの遅延比（実旅行時間 / 自由流旅行時間）を色で表示:



CC: 全リンクが緑（遅延比 ≈ 1.0 ） \rightarrow ネットワーク全体が自由流で動作

DD: 特定リンクが赤/オレンジ（遅延比 > 1.5 ） \rightarrow ボトルネックが局所的に発生
定周期信号 + UE経路選択の組合せが特定リンクへの交通集中を引き起こす

PD条件の検証

実験値からPD成立に必要なDRL導入コスト c の範囲を算出:

$$\Delta_{\text{coop}} = T_{DC} - T_{CC} = 139.4 - 121.2 = 18.2 \text{ s}$$

$$\Delta_{\text{selfish}} = T_{DD} - T_{CD} = 174.1 - 129.2 = 44.9 \text{ s}$$

$44.9 < c < 52.9 \text{ s}$ のとき囚人のジレンマが成立

$$\text{PD領域の幅} = T_{CD} - T_{CC} = 8.0 \text{ s}$$

信号制御者の費用マトリクス ($c = 49 \text{ s}$ の例) :

	ドライバー SO	ドライバー UE
DRL	$121.2 + 49 = 170.2$	$129.2 + 49 = 178.2$
Fixed	139.4	174.1

Fixed が支配戦略 ✓ CCがDDをパレート支配 ✓ → **PD成立**

ゲーム構造の相転移

DRL導入コスト c の増加に伴いゲーム構造が変化する:

コスト範囲	Nash均衡	均衡ATT	ゲーム構造
$c < 18.2 \text{ s}$	CD (DRL, UE)	129.2 s	DRL支配
$18.2 < c < 44.9 \text{ s}$	CD (DRL, UE)	129.2 s	非対称
$44.9 < c < 52.9 \text{ s}$	DD (Fixed, UE)	174.1 s	囚人のジレンマ
$c > 52.9 \text{ s}$	DD (Fixed, UE)	174.1 s	CCも非効率

臨界点 $c^* = 44.9 \text{ s}$ を超えると Nash均衡が CD \rightarrow DD に遷移

ATTが $129.2 \text{ s} \rightarrow 174.1 \text{ s}$ \rightarrow **不連続に34.7%悪化**

DRL投資がわずかに割に合わなくなるだけで、システム全体が劇的に劣化する

考察

PD領域幅 = $T_{CD} - T_{CC}$ の意味

- DRLがドライバーの裏切りに**頑健**であるほど ($T_{CD} \rightarrow T_{CC}$)、PD領域は狭くなる
- DRLが完全に適応 \rightarrow PD消失 \rightarrow **DRLの技術的成熟がジレンマを解消**
- $CD_{\text{adapted}} \approx CD$ ($p = 0.67$): 現在のDRLは既にドライバー行動に頑健

DRLの非対称的改善効果

- SOドライバーに対する改善: $\Delta_{\text{coop}} = 18.2 \text{ s}$ (-13.1%)
- UEドライバーに対する改善: $\Delta_{\text{selfish}} = 44.9 \text{ s}$ (-25.8%)
- UEの不均衡なパターンに対してDRLの適応力がより発揮される

Price of Anarchy

$$\text{PoA} = T_{DD}/T_{CC} = 174.1/121.2 = 1.44 \quad (44\% \text{の効率損失})$$

まとめ

1. **PD成立条件を導出:** DRL導入コスト c に対し $T_{DD} - T_{CD} < c < T_{DD} - T_{CC}$ でPD成立
本実験: $44.9 < c < 52.9$ s
2. **PD領域幅** = $T_{CD} - T_{CC}$: DRLの対UE頑健性を反映
DRL技術の成熟 → PD領域消失 → ジレンマ解消の可能性
3. **臨界点 c^* での不連続的悪化:** DRL投資の費用対効果がわずかに負になるだけで ATT +34.7%
4. **PoA = 1.44:** 相互裏切りは相互協調に比べ44%の効率損失

示唆: DRL信号制御の**技術的性能だけでなく導入コスト**がゲーム構造を決定する
→ 協調メカニズム設計において**DRL導入コスト低減が本質的に重要**

- 以上の研究は、AIが半日でやりました
 - Claude Opus 4.6 on Claude Code (before 1M context)
- 最初のお題は人間が入力
 - 抜粋：交通信号の深層強化学習に関するシミュレーション研究をしてください。信号制御における信号制御vsドライバーの囚人のジレンマ・社会的ジレンマ構造をテーマとし、それに関するものであればなんでもかまいません。
- その後AIが半自律的に以下を実行
 - モデル構築・数理解析
 - コーディング
 - 数値実験
 - 結果分析
 - レポート作成
 - スライド作成
- 人間はAIの試行錯誤中に時々指示を出しただけ。コードは一行も書いてない。コマンドも実行してない。出力されたコードや計算結果の全貌も把握していない

- AIエージェントによる研究事例紹介
 - 「DRL信号制御とドライバー経路選択における囚人のジレンマの成立条件」
 - AIエージェントとは
 - 実際の研究の流れ
 - 知見

- AIを用いた交通研究・教育の今後
 - 研究
 - 大学・大学院教育

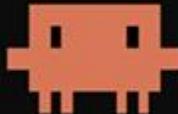
- 質問：AIの日常的な利用経験は？（試しに1回触った、みたいなのは除く）
 1. Webブラウザ上での対話型AI
 - 例：ChatGPT, Claude, Gemini
 2. API経由でクラウドLLMをコード・ツールに組み込んで使用
 3. 自分のPCでローカルLLMを動かした
 4. コーディング向け・CLI環境メインのAIエージェント
 - 例：Cline, Claude Code, Codex CLI, Antigravity
 5. Webブラウザ・PC自動操縦系のAIエージェント
 - 例：OpenClaw
 6. LLM自体を自分で訓練・fine-tuneしたことがある

- AIエージェント（コーディング・CLI環境向け）
 - 人間がターミナルで実行する
 - 人間がターミナル上で自然言語で指示を出す
 - ターミナルコマンドを自分の判断で実行できる
 - コマンド実行権限・読み書き権限は人間が管理できる
- コード作成→コード実行→デバッグ・改良→結果のレポート作成といった一連の流れを半自動的に実行できる。人間がやるのは言葉で指示するだけ
- 本体はChatGPT等と同様のLLMで、その入出力と実行ループを工夫している
- 代表例
 - Claude Code
 - Codex CLI (OpenAIのGPTベース)
 - 普通はサブスク（20~200ドル）。20ドルでも結構使える
 - 更にwebブラウザ操作などのPC操作をほぼ全て自動操縦できるモノもあるが、危険もあるので分別が必要
 - 研究向けの自律AIもでてきてるらしい

人間
入力→

```
PS C:\Users\... \Dropbox\pyworks\test_traffic_assignment> claude
Claude Code v2.1.77

Welcome back TS!



Opus 4.6 (1M context) · Claude Max
~\Dropbox\pyworks\test_traffic_assignment

Tips for getting started
Ask Claude to create a new app or clone a repository

Recent activity
No recent activity
```

人間
入力→

```
↑ Opus now defaults to 1M context · 5x more room, same pricing
> 静的交通量配分の簡単なアプリをpythonで作って
● まず現在のディレクトリの状態を確認します。
● Read 1 file (ctrl+o to expand)
● 空のディレクトリですね。静的交通量配分 (User Equilibrium: 利用者均衡配分) のアプリを作成します。
Frank-Wolfe法を使ったBPR関数ベースの利用者均衡配分を実装します。
· Scampering... (31s · ↓ 85 tokens)
```

```
def main():
    print("=" * 60)
    print("静的交通量配分（利用者均衡配分）")
    print("Frank-Wolfe法 + BPR関数")
    print("=" * 60)
    print()

    network = create_sample_network()

    print("ネットワーク情報:")
    print(f"   ノード数: {len(network.nodes)}")
    print(f"   リンク数: {len(network.links)}")
    print(f"   ODペア数: {len(network.od_pairs)}")
    print()

    flows = frank_wolfe(network, max_iter=200, tol=1e-5)
    print_results(network, flows)

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

人間
入力→

```
-----
Do you want to create traffic_assignment.py?
> 1. Yes
   2. Yes, allow all edits during this session (shift+tab)
   3. No

Esc to cancel · Tab to amend
```

```
... +241 lines (ctrl+o to expand)
```

- 動作確認します。
- Bash(`python traffic_assignment.py`)
└ Running...

Bash command

```
python traffic_assignment.py  
Run traffic assignment app
```

This command requires approval

Do you want to proceed?

- › 1. Yes
- 2. Yes, and don't ask again for: python:*
- 3. No

Esc to cancel · Tab to amend · ctrl+e to explain

人間
入力→

- Bash(`python traffic_assignment.py`)

```

=====
❖I [z i t z
Frank-Wolfe + BPR
... +27 lines (ctrl+o to expand)

```

- 計算は正しく動作しています（3回で収束）。Windowsコンソールで日本語が文字化けしていますが、結果は正しいです。

結果の解釈：

- 5リンク・4ノードのサンプルネットワークで、ノード1→4に需要6を配分
- 直行ルート（Link 4: 1→4）に約5.68、迂回ルート（Link 1,3: 1→3→4）に約0.32が配分
- 両経路の旅行時間が約15.0で等しい → **利用者均衡（Wardropの第一原則）**が成立

アプリの機能

機能	内容
配分手法	Frank-Wolfe法（凸結合法）
リンクコスト関数	BPR関数（ $\alpha=0.15$, $\beta=4.0$ ）
最短経路	Dijkstra法
ステップサイズ	黄金分割法
収束判定	相対Gap

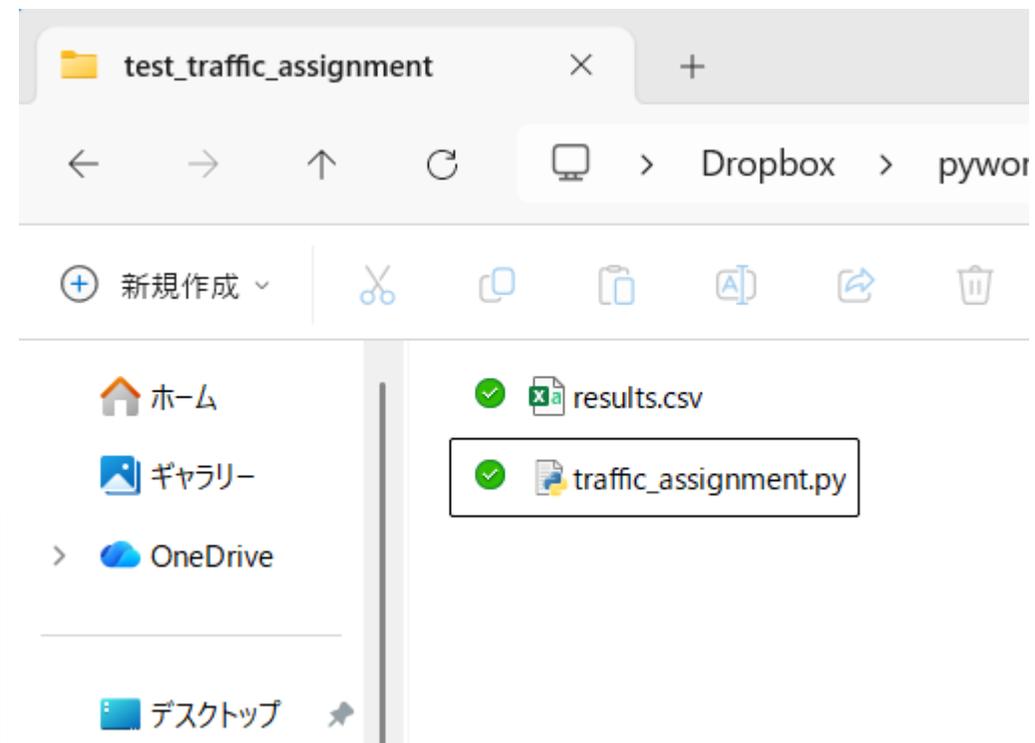
人間
入力→

```
結果をcsvに保存するようにして
```

- Read 1 file (ctrl+o to expand)
- Update(traffic_assignment.py)
 - └ Added 1 line
 - 3 Frank-Wolfe法 + BPR関数によるUser Equilibrium配分
 - 4 """
 - 5
 - 6 +import csv
 - 7 import numpy as np
 - 8 import heapq
 - 9 from dataclasses import dataclass, field

● CSV保存できました。results.csv に以下のカラムで出力されます：

カラム	内容
link	リンク番号
origin / destination	起終点ノード
flow	配分交通量



- ターミナルとテキストファイル編集でできることは何でもできる
 - コーディングに限らない
 - 議事録整理, 論文執筆とかも
 - 画像など他のフォーマットのファイル, GUIも場合によっては扱える
- 権限管理は人間の責任で, 自分が何をやらせているのかは理解しないといけない
 - クラウドLLMを使う場合, AIが読んだ内容はプロバイダに全てアップロードされる!
 - コードが読んだ内容はAIには読めない(表示しない限り)ので, 注意深く分離することは可能
- 大雑把な略史
 - (前史: 2021/6 コード補間AI GitHub Copilot登場)
 - (前史: 2022/11 対話型AI ChatGPT登場)
 - 2024年からAIエージェントの登場がはじめる
 - Cursor, Devin, Cline (Roo Code)など
 - 2025年に大手からClaude Code, Codex CLIが出る
 - 未来を感じるソフトではあったが, LLMのハルシネーションやサボり癖もあり粗は目立った
 - 2026/2にClaude Opus 4.6がリリースされ, これを使うClaude Codeの信頼性が非常に高くなる
 - Codex CLIもGPT-5.4ベースになり, 噂では思考レベルが高いらしい

- まず、人間から大まかなテーマを指定され、それに基づく研究計画作成を指示される

> 交通信号の深層強化学習に関するシミュレーション研究をしてください。信号制御における囚人のジレンマ・社会的ジレンマがテーマで、それに関するものであればなんでもかまいません。交通シミュレータとしてはPythonのUXsimを用い、python (+ pytorchなど) で完結するものとしてください。プロトタイプینگなので、簡易な研究でかまいません。まずは計画を立ててください。

■ AIが研究計画を提示する（長いので抜粋）

交通信号DRL制御における社会的ジレンマの分析

Context

交通信号制御には、信号制御者（システム最適化を目指す）とドライバー（個人の移動時間最小化を目指す）の間に囚人のジレンマ構造が存在する。信号がシステム最適な制御を行っても、ドライバーが利己的に経路選択するとその効果が損なわれる。本研究では、UXsimとPyTorch DQNを用いて、この社会的ジレンマ構造を定量的に示す。

実験設計

2×2ゲーム構造

	ドライバー協調	ドライバー利己的
信号 協調 (DRL)	CC: 最良	CD: 信号の努力が裏切られる
信号 利己的 (Greedy)	DC: ドライバーが搾取される	DD: 最悪

DRL (DQN) 設計

独立エージェント（パラメータ共有）方式:

- 各信号交差点に1エージェント、同一ニューラルネットを共有
- 状態 (per intersection): 各流入リンクの車両数(正規化) $\times 4$ + 待ち行列数(正規化) $\times 4$ + 現在フェーズ(one-hot) $\times 2$ + フェーズ経過時間(正規化) $\times 1$ = **11次元** (中央, 辺ノードは9次元→11にゼロパディング)
- 行動: フェーズ0(EW青) or フェーズ1(NS青) → 2択
- 報酬:
 - 協調: **$-\text{sum}(\text{全リンクの待ち行列車両数})$** (グローバル)
 - 利己的: Greedy制御なのでDRL不使用
- 制御間隔: 10秒ごとに行動選択

DQNハイパーパラメータ:

- ネットワーク: MLP (11→64→64→2)
- 学習率: $1e-3$, γ : 0.95, ϵ : 1.0→0.05 (decay 0.995/episode)
- バッチサイズ: 32, リプレイバッファ: 10000
- ターゲット更新: 10エピソードごと
- 訓練: 200エピソード

■ AIが研究計画を提示する（長いので抜粋）

実験手順

1. **DRL訓練（CC条件）**：協調ドライバー環境でDRL訓練 → 200エピソード
2. **DRL訓練（CD条件）**：利己的ドライバー環境でDRL訓練 → 200エピソード
3. **評価（4条件×20シード）**：
 - CC: 訓練済みDRL(CC) + 協調ドライバー
 - CD: 訓練済みDRL(CC) + 利己的ドライバー（協調前提のDRLが裏切られる）
 - DC: Greedy信号 + 協調ドライバー
 - DD: Greedy信号 + 利己的ドライバー
4. **追加評価**: CD訓練済みDRL + 利己的ドライバー（適応型DRL）
5. **分析・可視化**

評価指標

- 総旅行時間 (TTT): `W.analyzer.basic_to_pandas()`
- 平均旅行時間 (ATT): 同上
- 総遅れ (Total Delay)
- 完了トリップ数
- **Price of Anarchy**: $ATT(DD) / ATT(CC)$
- ペイオフマトリクス: 2×2 の平均旅行時間

ファイル構成

```
traffic_signal_experiments/  
├── requirements.txt      # 依存パッケージ  
├── config.py            # 全ハイパーパラメータ・実験  
設定  
├── traffic_env.py      # UXsim RLラッパー環境  
├── dqn_agent.py        # DQN実装 (PyTorch)  
├── run_experiment.py   # メイン: 訓練・評価・分析・  
プロット
```

各ファイルの役割

1. **config.py**: ネットワーク・需要・DRL・実験の全パラメータを一元管理
2. **traffic_env.py**: UXsimをGym-likeインターフェースでラップ
 - `reset()`: World生成、ネットワーク構築、需要設定
 - `step(actions)`: 信号フェーズ設定→シミュレーション実行→状態・報酬・完了判定
 - `driver_mode` パラメータで協調/利己的を切替
3. **dqn_agent.py**: 標準DQN（リプレイバッファ、 ϵ -greedy、ターゲットネットワーク）
4. **run_experiment.py**: 全実験の実行・結果集計・図表生成

- 流れ
 1. 人間がテーマを提示
 2. AIが計画を立て，人間が承認
 3. AIがコード作成，実行，結果の分析，コード改良を半自律的に進める
 4. ひと段落したら，AIに結果をレポート・スライドにまとめさせる
 5. 人間がレビュー．大きな問題があれば直すよう指示
 6. これを繰り返し，人間が満足したら終了
- 今回研究の分析時では，人間の指示は合計25件．ほとんどは1行短文
 - スライド調整時に細かいやり取りが多数発生したが，それはカウント外
- 人間の役割は，手の動く学生RAに片手間で口頭で指示する指導教員という感じ

実際の人間の入力（全部）

- 人間の入力一覧
 - 当時は他人に見せるつもりはなく入力したのだが、ログが残っていることがわかったので開示。雑なものはご容赦ください
- タイムスタンプは世界標準時

いきなり実装するのではなく、計画を立てさせるコマンド

AIに独自知識を付与するSkillsという機能を使ったつもりだったが、使い方が間違っていてたぶん意味ない

約15分かけて研究計画を練る

[02:52:51] /plan

[02:53:28] ここは空のディレクトリです。ここで、交通信号の深層強化学習に関するシミュレーション研究をしてください。信号制御における信号とドライバー（社会）の間にある囚人のジレンマ・社会的ジレンマ構造をテーマとし、それに関するものであればなんでもかまいません。交通シミュレータとしてはPythonのUXsimを用い、python（+pytorchなど）で完結するものとしてください。プロトタイピングなので、簡易な研究でかまいません。まずは計画を立ててください。UXsimをつかうcode skillsは\uxsimとして定義されています。

[03:15:28] 仮想環境は既にアクティベートされているから、pipとかpythonを直接呼べるはず。一応確認して

[03:27:54] deha

[03:28:00] では実験を実行してください

初回のシミュレーションはエラーもなく無事に回ったが、意図した結果が出なかった

[03:34:36] 修正して再実験してください。今後、意図した結果がでるまで自由に試行錯誤してください。こちらの許可は不要です

放置して数時間経過

色々やった後、学習が不安定なDRLではなく、信頼性の高いWebsterを使いたいという提案をして止まっていた

[06:02:54] AIを使うのが重要なポイント。DRLを使うこと

非常に真っ当な提案ではあるのだが、お題から外れるので却下し別の提案を返す

[06:03:16] Greedyの方を単なるfixed鄭州気にしたら

[06:03:22] 定周期

[06:23:27] 協調ドライバーは渋滞を起こさないよう均等に経路を分散しておくとかは？

自作シミュレータUXsimの機能を教える
(逆に言えば、ここ以外では教えておらず、ドキュメントとソースコードを読んで自分で勉強してる)

[06:29:24] duo_noiseはそういう機能ではないよ。Vehicle.enforce_routeで個別に設定できる

[06:32:48] 前回のエージェントが走ったままじゃない？

[06:33:13] あと、コンテキストがあふれそうだから、今切りのいい状態でリセットして

[06:34:28] /compact

[06:35:52] v10の実験を実行して

囚人のジレンマになりました！
という報告が上がってくる

[06:51:32] 素晴らしい！レポート形式でmdにまとめてください

[06:56:29] 6ページスライドにして。学会の短いプレゼンで、背景＋既往研究、目的、方法、実験設定、結果、まとめ

Marp: Markdownテキストベースのプレゼンフォーマット

[06:56:46] marpで

[06:57:54] 方法や実験設定では、数式や技術的なことも述べて高度なことをしている感を出して

レポートを見ると実は囚人のジレンマになってなかったのでキレル

AIは素直に認め、問題点を整理する

AIが自分で数式解析し、「囚人のジレンマになるのはパラメータがこの範囲の場合だけなので、ジレンマになりにくいです」と言ってくる

大発見というわけでは全くないが、AIオリジナルの貢献と評価できる

せっかくなので、ちょっと強引だが、AIの貢献を生かした研究に繋げてみる

→ **[07:14:19]** これって囚人のジレンマなの？

[07:15:24] ドライバーは個人で見ればUE行動をしているということなので、非協力でよいでしょう。一方、信号は修正が必要です。「DRL制御を導入するコスト」が追加でかかるということにすればどう？

→ **[07:22:30]** 信号のコストで、囚人のジレンマになる幅があるというのは非常に良い知見です。それは明らかに本研究の新発見です。このゲーム理論としての理論の修正と幅の発見に基づき、レポートを完全に書き直してください。前回からの差分ではなく、全く新しい自己完結したレポートにしてください

[07:23:06] ドライバー側は現在の扱いでOKです。UEとSOはそうなるというのはすでに学術的に共通認知があるので

結果が出そろったので、スライドに
まとめさせる

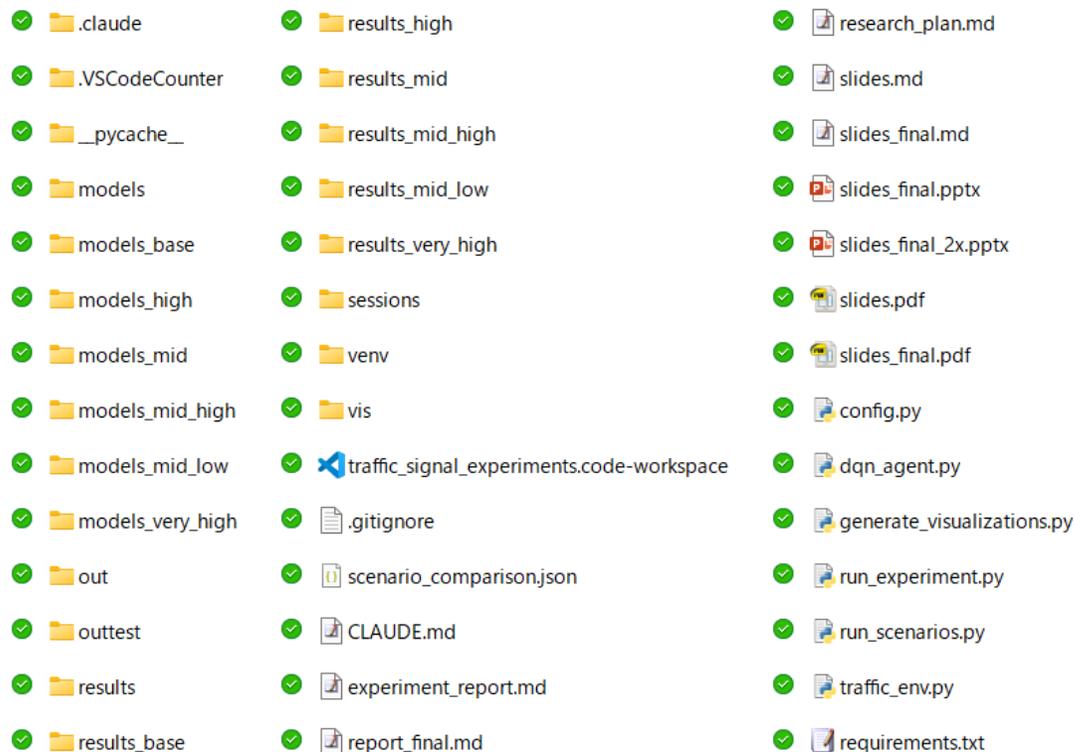


[07:32:09] この結果に基づき、marpのプレゼン資料をつくってください。ただし、前回みたいな二段組は止めてください。そのかわりに10ページくらいで

[07:36:40] シミュレーションでUXsimの説明もしておいて。その長所を生かした応用例であるのもポイントなので

この後、スライドの文字がはみ出るとか可視化が見にくいとかの細かい見栄えのやり取りが20件程度続く。このログ抽出と整形も当然claude codeにやらせた

- 構造化されたプロジェクトディレクトリが出来上がる
- コードはPythonで8ファイル, 合計1909行
 - 中身は普通に丁寧なAIらしいコード
 - きちんと瀬尾製シミュを使いこなしてるのがすごい



```
def _build_network(self):  
    """Build 3x3 grid network with signals at interior nodes."""  
    W = self.W  
    self.nodes = {}  
    self.links = {}  
  
    # Create nodes  
    for i in range(C.GRID_SIZE):  
        for j in range(C.GRID_SIZE):  
            coord = (i, j)  
            if coord in C.SIGNAL_NODES:  
                if self.signal_mode == "fixed":  
                    # Fixed-time: UXsim built-in cycling, no manual override  
                    node = W.addNode(f"{coord}", i, j,  
                                     signal=C.FIXED_SIGNAL_PHASES)  
                else:  
                    # DRL/Greedy: very long phases, manual override via signal_phase  
                    node = W.addNode(f"{coord}", i, j, signal=[9999, 9999])  
            else:  
                node = W.addNode(f"{coord}", i, j)  
        self.nodes[coord] = node
```

- レポートは全7000字. そのまま論文にできるものではないが, ベースにはなる
- プレゼンスライドはさっき示したもの

DRL信号制御とドライバー経路選択における囚人のジレンマの成立条件

1. はじめに

交通ネットワークにおいて、信号制御者とドライバーはそれぞれ異なる目的関数を持つ意思決定主体である。深層強化学習（DRL）による適応的信号制御は定周期信号を上回る性能を示すが、DRL導入にはセンサ設備・計算資源・ML人材等のコストが伴う。一方、ドライバーは個人の旅行時間最小化を目指し、利己的な経路選択（User Equilibrium: UE）を行うのが支配戦略である（Wardropの第一原則）。

本研究では、この2つの意思決定を2×2ゲームとして定式化し、**DRL導入コスト c** をパラメータとして囚人のジレンマ（PD）の成立条件を理論的に導出する。さらに、メソスコピック交通流シミュレータUXsimとDouble DQNによる実験でPD成立を定量的に検証する。

2. ゲーム理論的定式化

2.3 PD成立条件の導出

Fixed（裏切り）が支配戦略となる条件:

- SOドライバーに対して: $T_{DC} < T_{CC} + c$ すなわち $c > T_{DC} - T_{CC} \equiv \Delta_{\text{coop}}$
- UEドライバーに対して: $T_{DD} < T_{CD} + c$ すなわち $c > T_{DD} - T_{CD} \equiv \Delta_{\text{selfish}}$

CC（相互協調）が**DD**（相互裏切り）をパレート支配する条件:

$$T_{CC} + c < T_{DD} \iff c < T_{DD} - T_{CC}$$

以上を統合すると、囚人のジレンマが成立する条件は:

$$\max(\Delta_{\text{coop}}, \Delta_{\text{selfish}}) < c < T_{DD} - T_{CC}$$

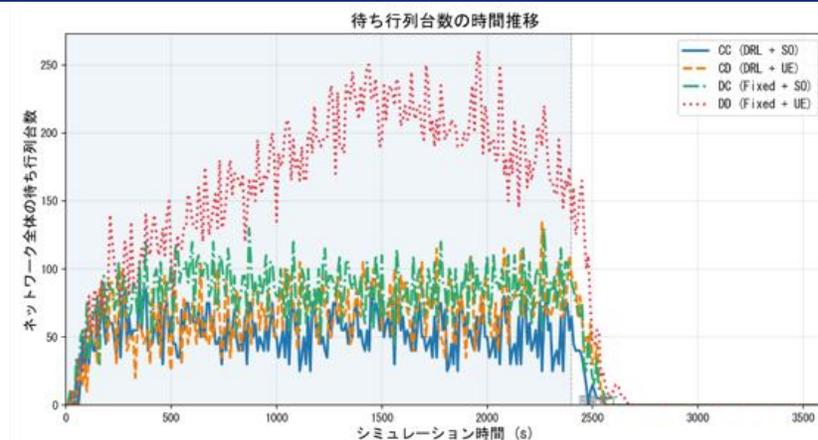
PD領域の幅は:

$$W_{\text{PD}} = (T_{DD} - T_{CC}) - \max(\Delta_{\text{coop}}, \Delta_{\text{selfish}})$$

$\Delta_{\text{selfish}} > \Delta_{\text{coop}}$ の場合（本実験の基本シナリオ）、 $W_{\text{PD}} = T_{CD} - T_{CC}$ に簡約される。これはDRL制御下でのドライバー逸脱コストに等しい。

- 粗は結構あり若干こじつけもあり，いきなり論文レベルとはいかないが，悪くないクオリティの仕事といえるのではないか
 - 特に半日の片手間の成果と考えると
 - AIで1週間くらいブラッシュアップすれば，どこかに論文として出せないこともないだろう
- 交通に関する直観も結構備えている
 - 適度に渋滞していると信号や経路選択の意味が大きくなるので，ちょうどいい需要を探すということをしている
 - 経路選択モデルのノイズを極端に大きくすれば全経路に均等に配分されるはず，という推察を勝手にしている
 - まずは深層強化学習よりWebsterを使うべき，という常識を持つ
- 無知の知 = 「自分にはわからないから教えてくれ」という判断がある程度できる
 - 「これはこのはず…」 「いやよく考えるとこう」 「ここはわからない」という自問自答ができる
 - 今回のケースでは，テーマに直結するハイレベルな意思決定はしないという選択をした
 - 対話を通じた探索的な研究に繋がられる．「インフラ整備費用に起因する囚人ジレンマ構造」というのは最初は人間側の頭にはなかった
 - 筋がいいか？と言われると微妙だが．．．
 - とはいえ，囚人のジレンマでミスをしていたのは大きな誤り．一任はできない

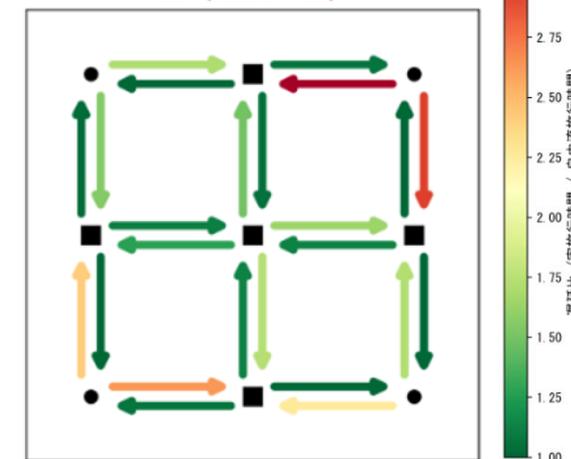
- 今回の研究はむしろAIにとって難易度が高いほうだろう
- AIにとって難易度の低い研究：モジュール化され，流れが確立している研究
 - 「{旅行時間}の{予測}のための{深層学習}手法に近年提案された{XXXアテンション}を組み込み，state-of-the-arts手法に比べて性能が上がりました」系の研究は無限にできる
- 研究の一部の具体的技術的タスクをやらせればよりうまくいく
 - 「いい感じにSPSAでパラメータチューニングして」
 - 「この最適輸送問題をシンクホーン化してpytorchのgpuで早く解けるようにして．元問題との誤差も感度分析して」
- Web検索や他ソフトとの連携もある程度できる．交通にありがちな画像，GUI，時空間認識はまだ微妙



↑ まあわかる

リンク別遅延マップ

DD (Fixed + UE)



↑ AI自作の努力は認めるがよくわからん．少なくともAIによる解釈は強引だった．たぶん実際はバンキング

- こちらの指示は、必要なキーワードさえ含めていれば、簡単に言うだけで文脈を理解してくれる
 - 人間側のドメイン知識は必須
 - 人間にできないことはAIにもできない
(一見できたように見えても、妥当かは全くわからない)
- 全てテキストベースで管理できるとベスト
 - 文書はtxtベタ打ちかMarkdown
 - 交通流シミュレータとしては、全てPythonコードで表現するUXsimはこの目的上最強！
 - 未知のツールでも、ソースコードやオンライン文書があれば使いこなしてくれる
 - データはCSVが素直。Python等で直読みできるデータベースも可。エクセルはちゃんとしたものなら読めなくはないが、非効率
- 個人情報などの守秘義務データがあるにつらい
 - データ確認→コード作成→実行→改良のループが自動で回せなくなる
 - 本物の生データは別フォルダに隔離し、AIにはダミーデータを触らせて開発するとか
 - データ分析部分だけローカルLLMという手もあるが、微妙
- 他にもAIエージェント固有の技術的テクニックは色々あるが、ここでは省略

[06:03:16] Greedyの方を単なるfixed鄭州気にしたら

[06:03:22] 定周期

AI研究の難しい点

- 成果を人間が咀嚼する時間がとれない
 - 普通の人間が1週間かけてやる仕事が、3分で上がってくる
 - 1行書いてエンターキー押すだけでやってくれる
 - いくらでも並列化できる
- コードは読んでいられない
 - 理論上は、人間が鉄の意志を持って必ずチェックすればいいはず。差分を抽出して読みやすくできるし、gitを駆使して開発過程を整理もできる
 - しかし、超高速で膨大な成果が上がってくるので、そんな気はすぐなくなる
 - プロジェクトを完成させるという観点からは、全部チェックするのはおそらく合理的でもない
- 根本的に異なる管理方法が必要
 - **結果の妥当性を簡単に確認できる方法**を用意し、**要所で確認**できるようにしたうえで、AIにいい感じに暴走させるスタイルがベストプラクティス？
 - この**要所での妥当性確認方法**が決定的に重要。自動化もありえるが、ぶっちゃけ直観に頼る部分が多い。この直観を持っている人しか使いこなせないのでは
 - おそらく、人間が技術に追いついておらず、アナログになっている段階
- ある意味でギャンブルそのものであり、危険もある

 最新版.docx

 最新版コピ-.docx

 最新版コピ-new.docx

 最新版確定版.docx

 最新版確定版_20260302.docx

 最新版最終.docx

 最新版最終_fix.docx

 最新版最終_fix_本当の最終.docx

旧世代の冗談

 run_exp3.py

 run_exp6_r4.py

 run_exp3_cp0.py

 run_exp6_r5.py

 run_exp3_new.py

 run_exp7.py

 run_exp3_r3.py

 run_exp7_new.py

 run_exp3_r4.py

 run_exp8.py

 run_exp3_r5.py

 run_exp8_new.py

 run_exp4.py

 run_experiments.py

 run_exp4_cp0.py

 run_grid_search.py

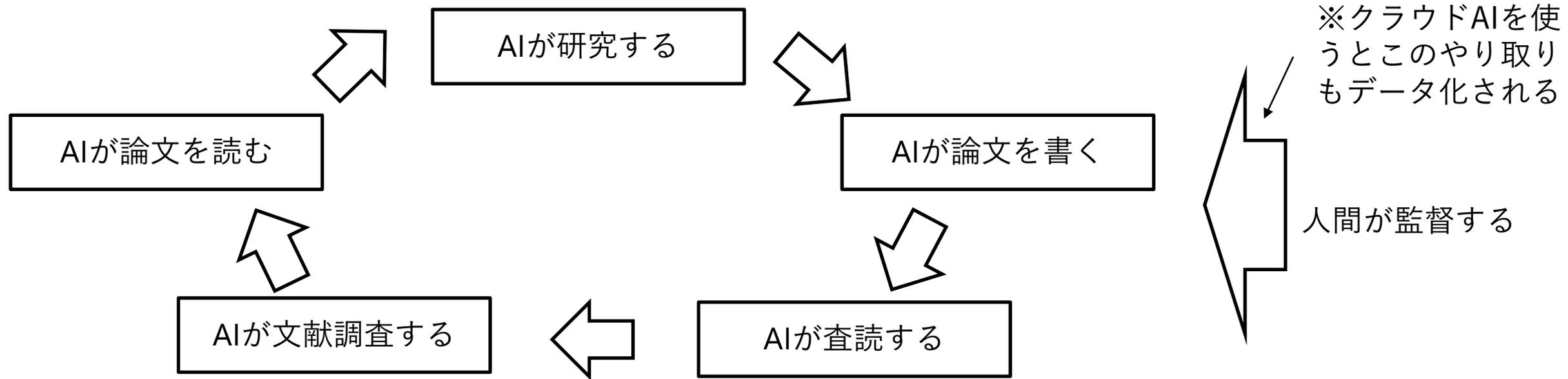
本当にこうなった

- 原則
 - AIはツールである
 - 人間がやるべきことをAIに丸投げしてはいけない
 - 人間は、AIが何をやっていることを全てを理解できなければならない
= 検証可能性を常に担保しなければならない
 - そのために、人間は原理原則を理解する必要がある
 - そのうえで、可能な時にAIを使って加速する
- この意味では、これまでのツールと変わらない。科学的アプローチでよい
- 机上作業における汎用性や、交通に対する直観的理解は大学院生RAレベル
- しかし、速度が人間の1000倍レベルで速い。これは質的な変化につながる

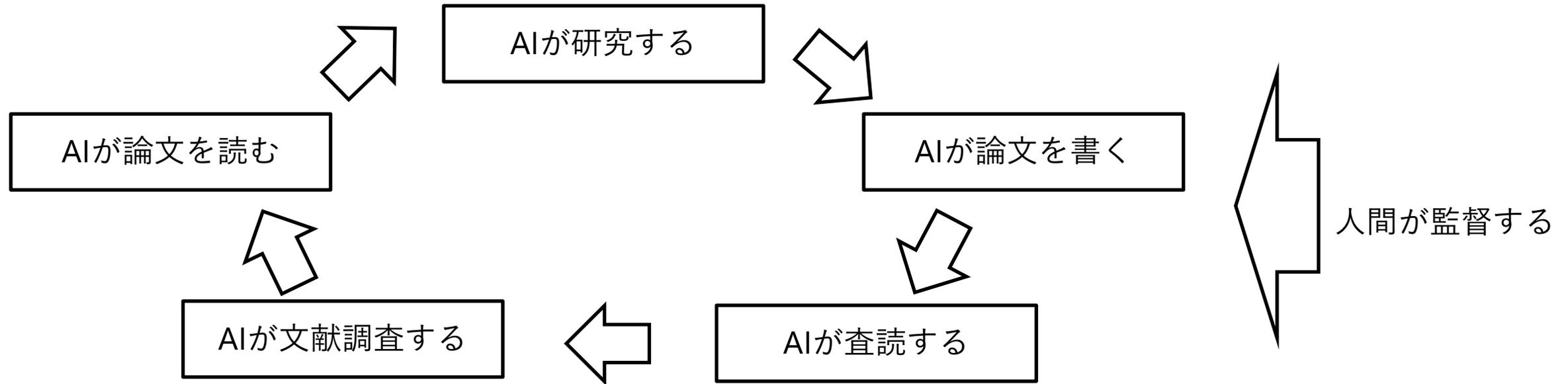
- AIエージェントによる研究事例紹介
 - 「DRL信号制御とドライバー経路選択における囚人のジレンマの成立条件」
 - AIエージェントとは
 - 実際の研究の流れ
 - 知見

- AIを用いた交通研究・教育の今後
 - 研究
 - 大学・大学院教育

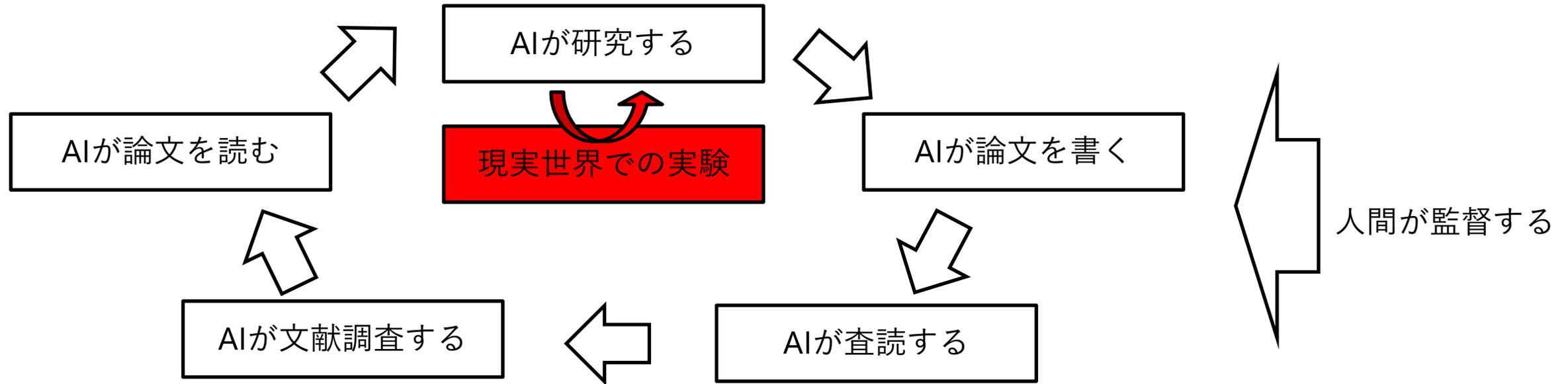
- 仕事でプログラムコードを人間が書くことは、近い将来にほとんどなくなる
 - 数学も少なくとも一部はコーディングの一種であり、同じことがおきるかも
- 問いを立てる能力，ドメイン知識，批判的思考能力，コーディング・数学スキルをすでに持っている人間は，AIを監督して使うことは難しくない
 - つまり，研究者タイプ
 - ほかに使いこなせるタイプはいるだろう
- スキルがある人間は今後どうすべきか？
 - 使えるものをすべて使って論文大量生産するのか？
 - 一週間で論文をAIに書かせて，ジャーナルに通るまで投稿しまくる．通らなくても大したダメージはない．それを並列でやる
 - やる人は絶対にいる．そうなるとうなるのか？



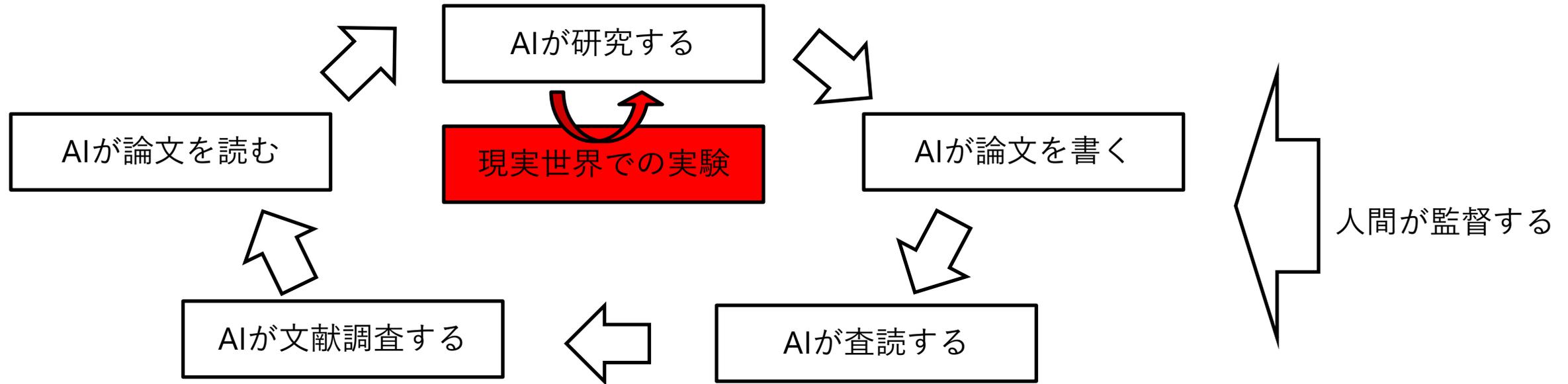
- これは我々の知っている研究ではないのでは???
- しかし、これを全否定するのは時代遅れの保守派なのかもしれない
- 「AI駆動型ジャーナル」的な場ができる？
 - データとコードも全部公開し、AI査読者が再現性や、都合のいい操作がないかも確認するみたいな
 - これはこれで技術的には健全
 - そういう場の試行は始まっているらしい：Agents4Science, <https://agents4science.stanford.edu/>
- それとScience/Nature的なEditor kickを多用する超厳格ジャーナルが並立する？
 - しかし、AI論文の飽和攻撃は人間には対処できない。AI査読はどこかには入れなければならない
 - これまでも査読者のクオリティは色々あったので大した違いではないかもしれない



- AI文献調査は技術的には一応これに追いつけそう
 - 現状でも、コンテキスト長1Mあれば論文100本が入る
 - サブエージェントを階層的に呼べば余裕だろう
- 出版社のペイウォール・オープンアクセス問題はどうなる？
 - AIにレビューしてもらえない論文は存在しないも同義になる
 - これまで同様著者がプレプリントを公開すればいいとはいえ、出版社も対応必要だろう
 - ゴールドオープンアクセス誌が主流になる？いい感じの収益分配モデルができる？



- 現実世界での実験が入ると、このループは極めて低速になる
- KPIが論文数だと、そのような研究の価値は極端に低くなる
- しかし、AIによる論文大量生産時代では、そういった研究こそ価値があるのではないか。どうやって測るか？
 - 引用数で自動的に測れるか？
- 分野ごとの論文数・引用数の価値がさらに変化する
 - ついに論文数がKPIでなくなる？



- 結局のところ短期的には、周りの目を気にせずに、各自が信じるところをやるしかないだろう。特に大学のような公的機関は囚人のジレンマに付き合うべきではない
 - AIで半自律的に新しい知見が得られて、業界がそれに合わせて変化できるなら悪くはない。粗製乱造論文スパムをきちんと排除できるか。人間が練習する場をどこに作るか
 - 任期あり教員はどうすべきか？質で勝負する少数の論文+そこそこの大量生産論文のミックス？
 - 「わかっている人」が質をちゃんと評価する必要。しかし、これとコネ社会の区別は難しい
- 業界がどう変わるか？を述べてきたが、それ以前に業界がAI変化に全く追いつけずに生じる歪みを心配するべきかもしれない。考えても非建設なことばかり思いついてあまり面白くないので論じなかったが、切実な問題かも

- 仕事でプログラムコードを人間が書くことは、近い将来にほとんどなくなる
 - 数学も少なくとも一部はコーディングの一種であり、同じことがおきるかも
- 問いを立てる能力，ドメイン知識，批判的思考能力，コーディング・数学スキルをすでに持っている人間は，AIを監督して使うことは難しくない
 - つまり，研究者タイプ
 - ほかに使いこなせるタイプはいるだろう
- スキルがない人間はどうすべきか？
 - 手作業でスキルを身に着けるか？
 - わからないままAI丸投げで使うべきか？
 - AIで効率よくスキルを身に着ける教育は可能か？
 - 教育される側の学生はどうすべきか？

- 産業革命で教育期間が伸びたのと似たような対応をすべきでは

- 身につけるべきスキルセット：問いを立てる能力，ドメイン知識，批判的思考能力，コーディング・数学スキル
- 上記とは独立に，AIの能力を把握し使いこなすスキル，誤りを見抜くスキルも重要なので，身につける必要
 - 研究室内AIコーディングコンペ＋ノウハウ共有とか楽しそう

- AIエージェントによる研究事例紹介
 - 「DRL信号制御とドライバー経路選択における囚人のジレンマの成立条件」
 - AIエージェントとは
 - 実際の研究の流れ
 - 知見

- AIを用いた交通研究・教育の今後
 - 研究
 - 大学・大学院教育

- 今日のAIは、歪ではあるが修士～博士課程学生RA並みの能力をもっている
 - 机上ではあらゆる分野の知識を持つ。人間の1000倍の速度で仕事をする
 - 物理的身体を持たない。常識と倫理感がない
 - テーマ提案レベルの価値判断は難しい

- 交通研究のツールとして
 - AIは机上作業であれば人間のRAとほぼ同じ感覚で使えるので、研究者は使いこなせる
 - 交通研究独自の要素：価値判断や現実世界との相互作用はAIには難しいので、当面はうまく共生できるだろう。守秘義務データの扱いは難しい
 - 人間の指導が不要になれば、我々はクビである。そうならないよう勉強する

- 研究はどうすべきか？
 - 遊びとしては大変面白い。アイデアを一瞬で具現化できる
 - 業績数が求められる仕事となると、色々な面でしんどい
 - 数ではなくて質を求めたい。一方で数からくる質もある。ゲームのルールは誰がどう決めるのか
 - 「査読付き論文」というものの在り方は、AI論文の飽和攻撃により大きく変わるはず
 - AIが再現性まで担保する「AI駆動型ジャーナル」はそれはそれで可能性がありそう
 - AIにできない研究（現実世界での実験など）の価値が本当は高まるはずだが、論文数がKPIの業界ではそのインセンティブが少なくなる。引用数で測れる？さしあたりは、個々の研究者が長期的な視野を持ってやるしかない？

- 学生の教育はどうすべきか？
 - 原則としては、これまでの原則を続ければよい。原理と演繹の理解、批判的思考
 - 実践は一筋縄ではいかない。AIをうまく使えば学習を加速できるので、バランスを考える必要