The Platform Design Problem

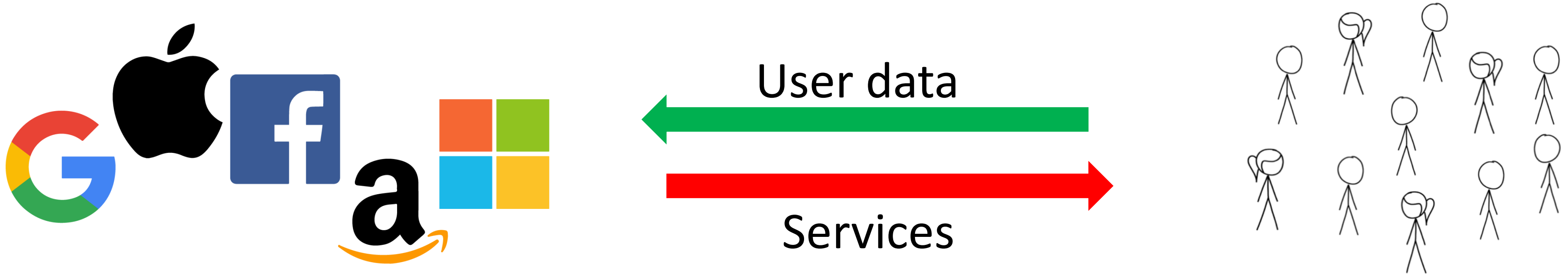
Christos Papadimitriou, **Kiran Vodrahalli**, Mihalis Yannakakis

Columbia University

WINE 2021

December 15, 2021

Economics of the Online Firm



Online firm

Users

- User data feeds revenue
  - Better demand segmentation
  - Ad/recommendation revenue
  - Better models => better services
- Online services bring value
  - Convenience
  - Knowledge

Outline

- Problem Definition
- General Case
- Tractable “Flower” Case
  - Agent Behavior
  - Designer’s Algorithm
- Extensions
- Summary
- Future Work

Problem Definition

Platform Design

Problem

Model the revenue-maximization problem of today's online firms (e.g. Google, FB, etc.) and understand computational tractability.

Bi-Level MDP Optimization Model

Agent: participates in Life MDP

Designer: tweaks the Life MDP by building platforms.

Goal: Designer wants to indirectly optimize its reward via Agent's optimal behavior! (Find Stackelberg)

- Key Idea: Google builds various apps (Maps, Search, Social Network, etc.) and profits based on usage of these apps.
- The usage of apps modifies the transitions of the Markov Chain of the user's life
- Assume the Designer has linear rewards over the steady state distribution of the resulting Markov chain (agent policy + Life MDP)

The Stackelberg Game

- Designer moves first:
  - Adds platforms which, if adopted, modify transitions to an existing Markov Chain
- Agent moves second:
  - Receives MDP from Designer, plays optimal behavior
- Example of bi-level MDP optimization
- What is the computational complexity of solving for equilibrium?

Formal Problem Statement

- An **agent** lives in an irreducible Markov chain with $A = [n]$ states.
- The **designer** chooses $S \subseteq A$ states to add platforms to.
- The agent may **adopt or not adopt** the platform at each state:
  - If **adopt**, the transitions change. Otherwise they do not.
  - Assume the chain remains irreducible.

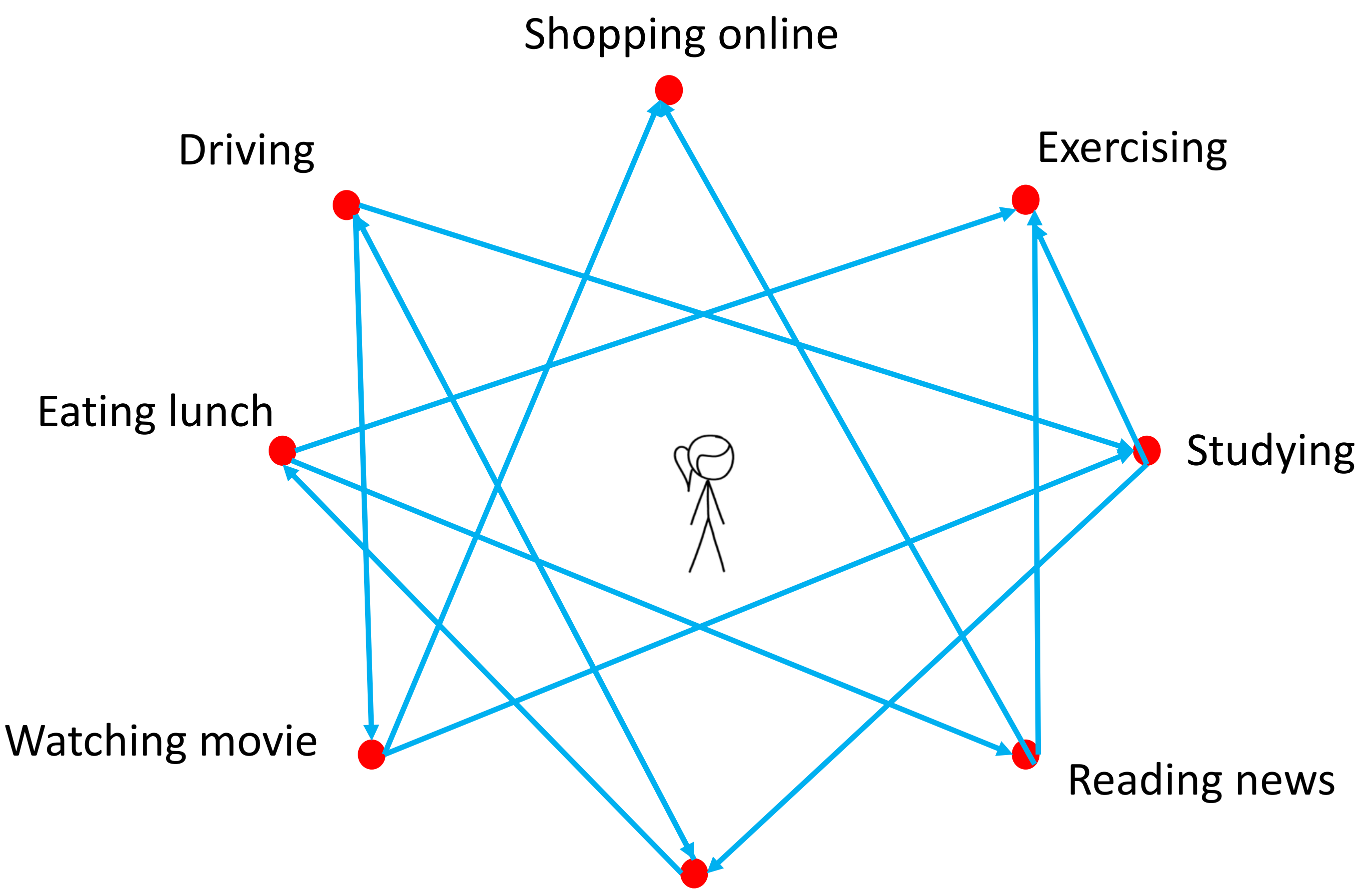
Formal Problem Statement

- Assign a utility rate for the agent (c_i) and the designer (d_i) at $i \in [n]$.
- The agent solves the resulting Markov Decision Process.
  - Resulting steady-state probabilities are given by π.
- The designer optimizes over S:

$$\mathrm{profit}(S) := \sum_{i \in S} d_i \cdot \pi_i(S) - \sum_{i \in S} \mathrm{cost}_i$$

General Case

Picture of the General Case

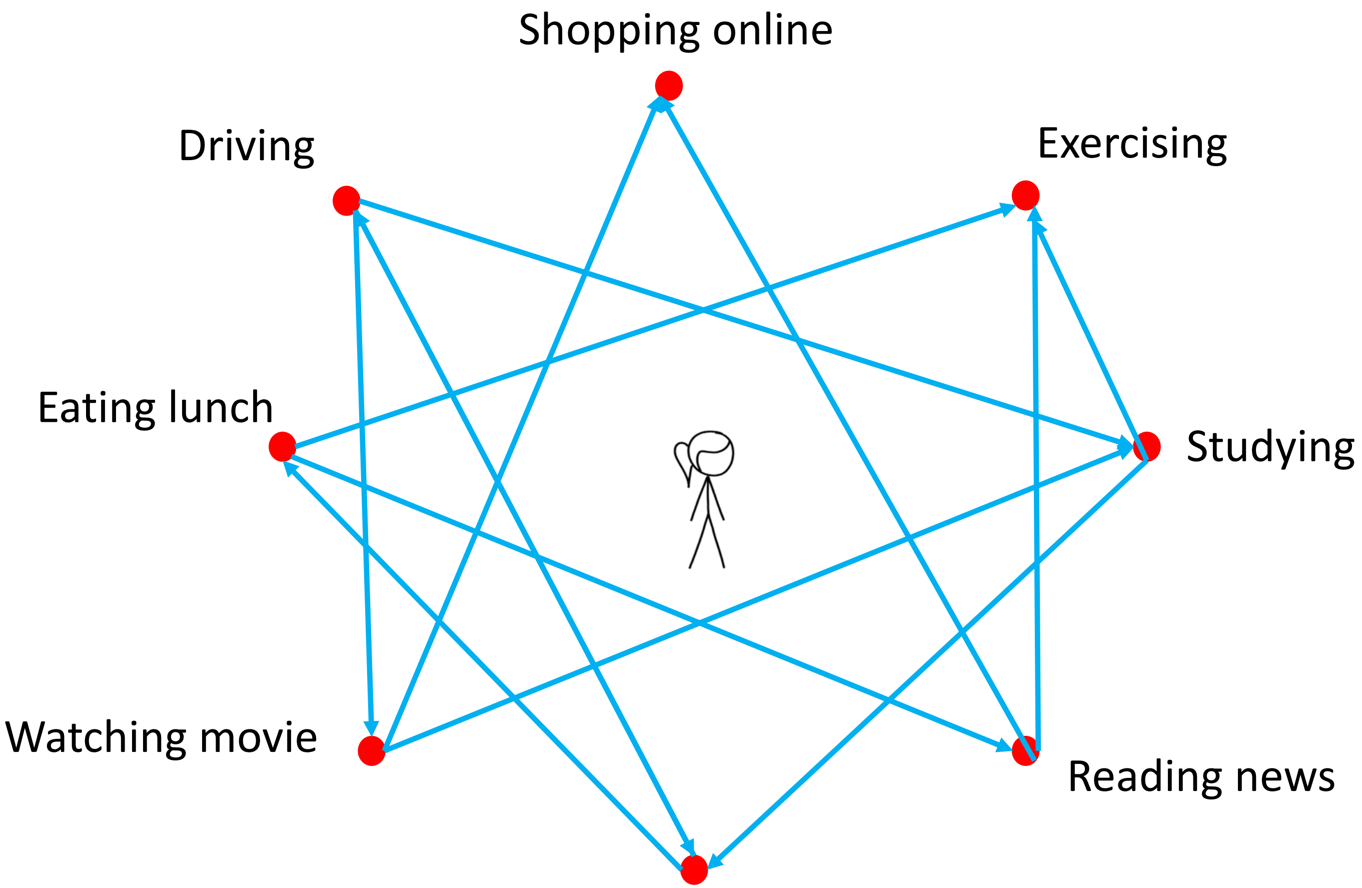


Agent's Life

What platforms should I build?

Online firm

Picture of the General Case



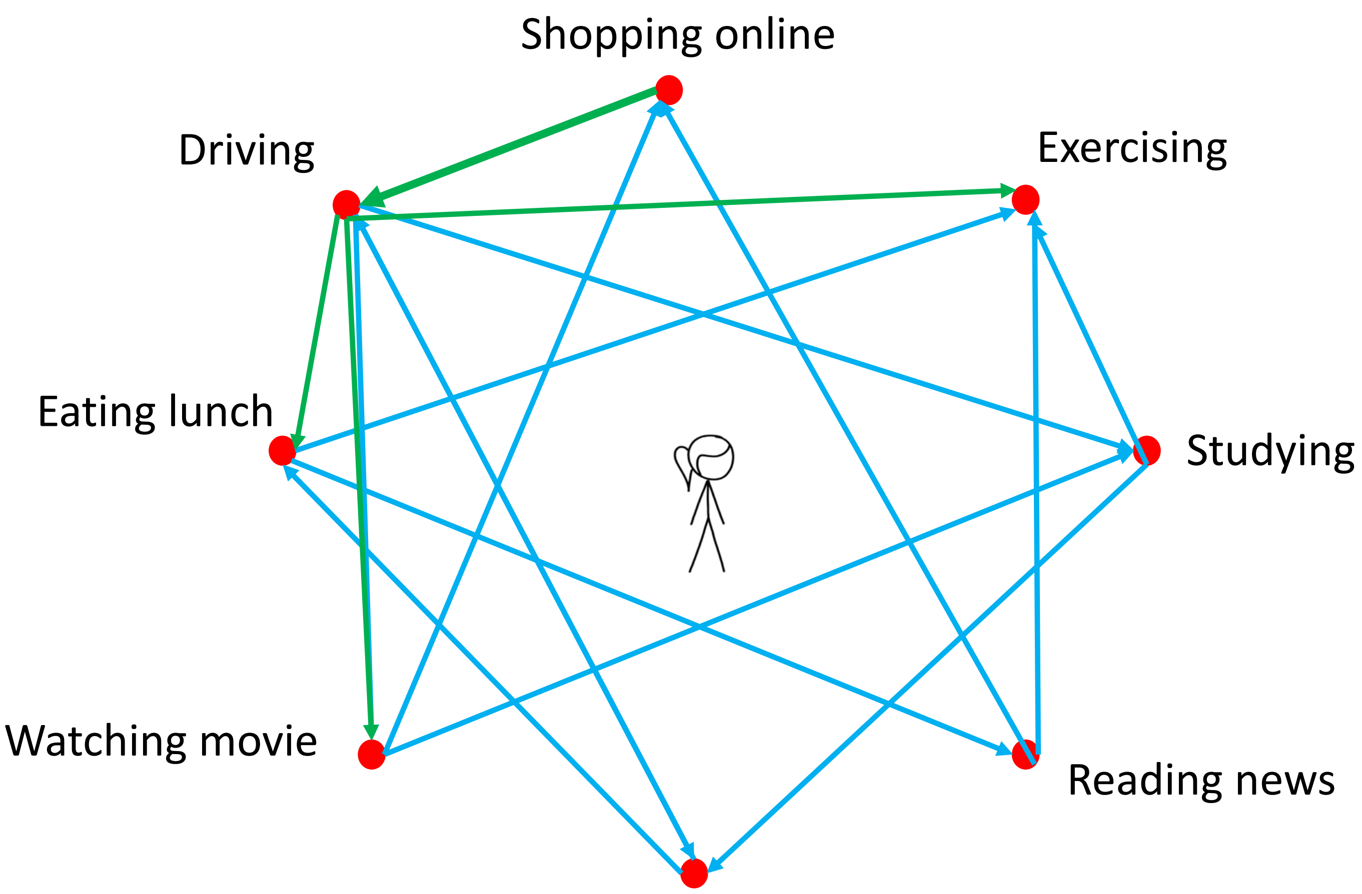
Agent's Life

What platforms should I build?

Online firm

At a cost, the firm can **add an opt-in action** to platforms they create (ex: Google Maps).

Picture of the General Case



Agent's Life
changes

Maybe we should create Maps technology….

Online firm

Builds platform Maps at a cost.

Opt in to Maps

Computational Tractability I: General Case

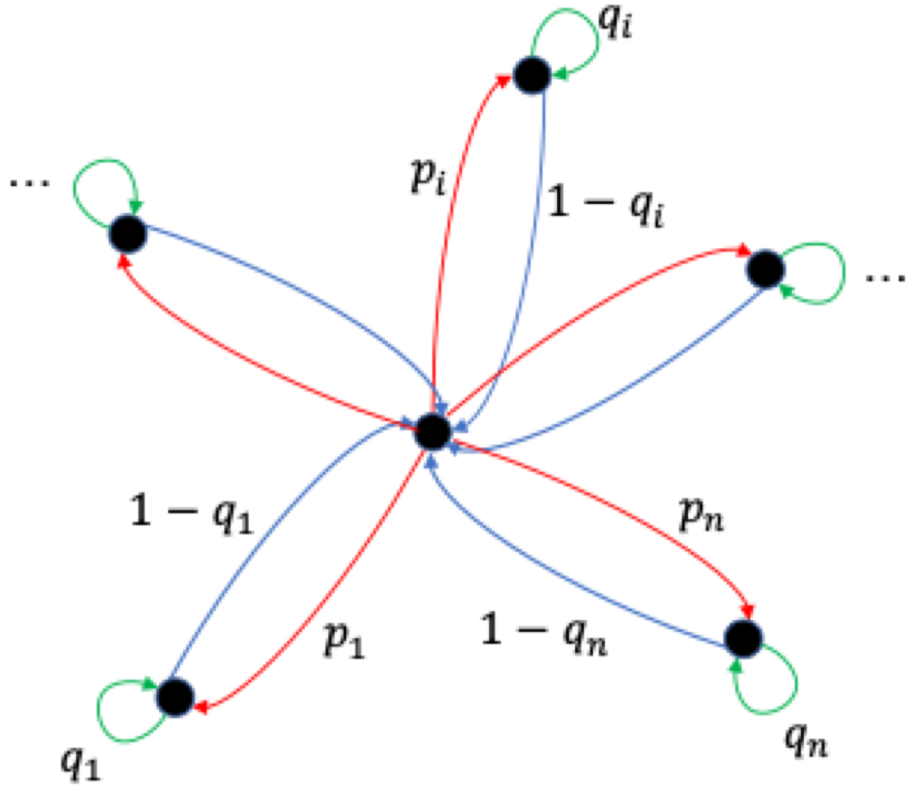
- It is strongly NP-hard to decide whether the Designer can obtain positive profit – and therefore hard to approximate.

- Reduction from SET COVER
  - Designer builds platforms which each solve subset of Agent's problems.
  - Most cost-effective covering set is NP hard.

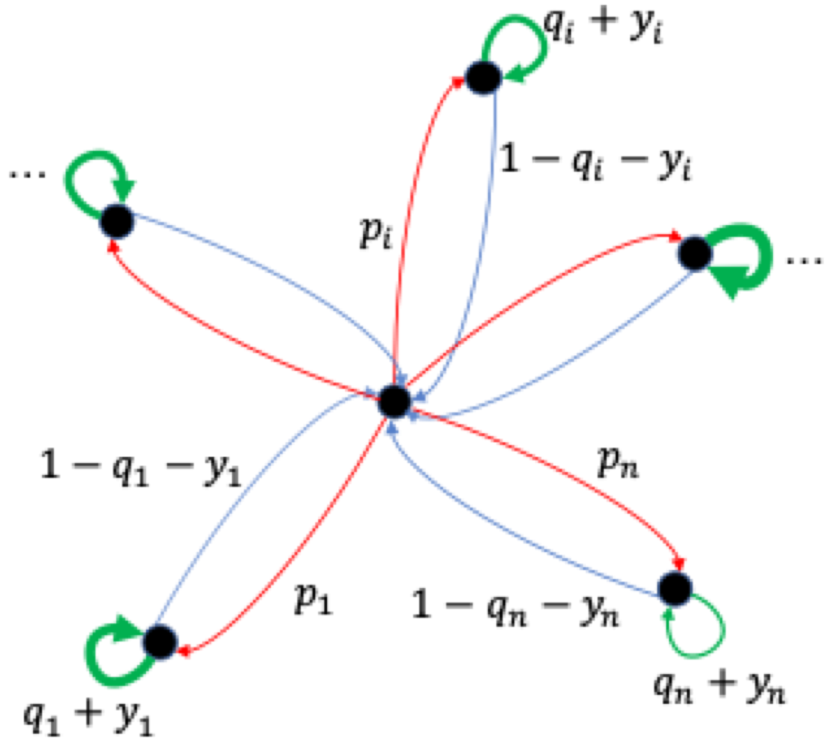
- In economic terms, the reduction exploits the complexity of "complementary goods."
  - Ex: Brick-and-mortar retail ads help the Agent discover the store, Maps helps the Agent get to the store.

Tractable “Flower” Case

A More Tractable Case: The Flower



Life MDP



Tweaked MDP via y_i

A More Tractable Case: The Flower

- Problem can be solved by an FPTAS

- Why tractable?
  - Substitutes rather than complements
    - Allocate time spent in each platform
  - Simpler low-level behavior (greedy agent)
  - Admits a DP upon discretization (knapsack DP)

Agent Behavior

The Agent's Greedy Algorithm

- Solving for the steady state distribution yields a quasi-concave combinatorial optimization problem:

Lemma 1. *The agent's objective for an optimal policy defined in Section $\boxed{2}$ can be re-written as the following optimization in the special case of the flower MDP (Definition $\boxed{2}$):*

$$\underset{S \subseteq [n]}{\operatorname{argmax}} \frac{A + \sum_{j \in S} z_j \phi(j)}{B + \sum_{j \in S} z_j} \tag{1}$$

where

$$A := \sum_{i=1}^{n} \lambda_i c_i^{\text{life}}; \quad B := 1 + \sum_{i=1}^{n} \lambda_i; \quad \lambda_i = \frac{p_i}{1 - q_i}; \quad z_i = \frac{p_i}{1 - q_i - y_i} - \frac{p_i}{1 - q_i} \geq 0;$$

$$\phi(i) := \begin{cases} c_i^{\text{platform}} + \frac{\lambda_i}{z_i}\left(c_i^{\text{platform}} - c_i^{\text{life}}\right) & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \end{cases};$$

We therefore define

$$\text{utility}^{\text{Agent}}(S) := \frac{A + \sum_{j \in S} z_j \phi(j)}{B + \sum_{j \in S} z_j}$$

The Agent's Greedy Algorithm

- Solving for the steady state distribution yields a quasi-concave combinatorial optimization problem:

Lemma 1. *The agent's objective for an optimal policy defined in Section 2 can be re-written as the following optimization in the special case of the flower MDP (Definition 2):*

$$\underset{S \subseteq [n]}{\operatorname{argmax}} \frac{A + \sum_{j \in S} z_j \phi(j)}{B + \sum_{j \in S} z_j} \tag{1}$$

where

$$A := \sum_{i=1}^{n} \lambda_i c_i^{\text{life}}; \quad B := 1 + \sum_{i=1}^{n} \lambda_i; \quad \lambda_i = \frac{p_i}{1-q_i}; \quad z_i = \frac{p_i}{1-q_i-y_i} - \frac{p_i}{1-q_i} \geq 0;$$

$$\boxed{\phi(i) := \begin{cases} c_i^{\text{platform}} + \frac{\lambda_i}{z_i}\left(c_i^{\text{platform}} - c_i^{\text{life}}\right) & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \end{cases};}$$

Potential function

We therefore define

$$\text{utility}^{\text{Agent}}(S) := \frac{A + \sum_{j \in S} z_j \phi(j)}{B + \sum_{j \in S} z_j}$$

The Agent's Greedy Algorithm

ALGORITHM 1: GREEDY ALGORITHM

Input: Parameters of the Agent's problem: transition probabilities and utility coefficients in and out of the platform.

Output: An optimal subset $S \subseteq [n]$ of states where the Agent accepts the platform.

```
Initialize S := {}
for k ∈ [n] sorted[9] from largest to smallest φ(k) do
    if utility^Agent(S) < φ(k) then
        Update S := S ∪ {k}
    else
        return S
    end
end
return S
```

Sort states by potential function and add until utility = potential

Designer’s Algorithm

Recall: Designer's Objective

$$\text{profit}(S) := \sum_{i \in S} d_i \cdot \pi_i(S) - \sum_{i \in S} \text{cost}_i$$

Set of states to build platforms

Designer's steady-state reward rates

Agent's steady state probabilities

Designer's one-time costs for building each platform

Restrictions

- Expanding the profit function given agent behavior:

$$\text{profit}(S) := \frac{\sum_{i \in \text{Agent}(S)} d_i \cdot \frac{p_i}{1-q_i-y_i}}{B + \sum_{i \in \text{Agent}(S)} z_i} - \sum_{i \in S} \text{cost}_i$$

$P_1(S)$ $D(S)$

- Define $\max_i d_i =: K$
  - Maximum profit is nK
- Assume z_i are $poly(n)$ and discretized with gap δ and costs are $K * poly(n)$

Target Algorithm

- Deciding whether it is possible to attain a certain profit is NP complete

- Reduction from PARTITION

- Thus, our goal: A $(1 - \epsilon)$ approximate algorithm in polynomial time.

The Designer's Dynamic Program

- **Key Idea:** Use a (poly-sized) hash table with rounded rewards
- Difficulty comes from profit scale and non-discretized z_i
- Hash function:

$$\mathrm{hash}(S) := \left(\lceil \frac{\mathrm{profit}(S)}{\epsilon K/2n} \rceil, \lceil \frac{P_1(S)}{\epsilon K/2n} \rceil, \mathbf{D}(S)/\delta \right)$$

- Similar to standard Knapsack FPTAS (Ibarra & Kim, 1975)

Extensions

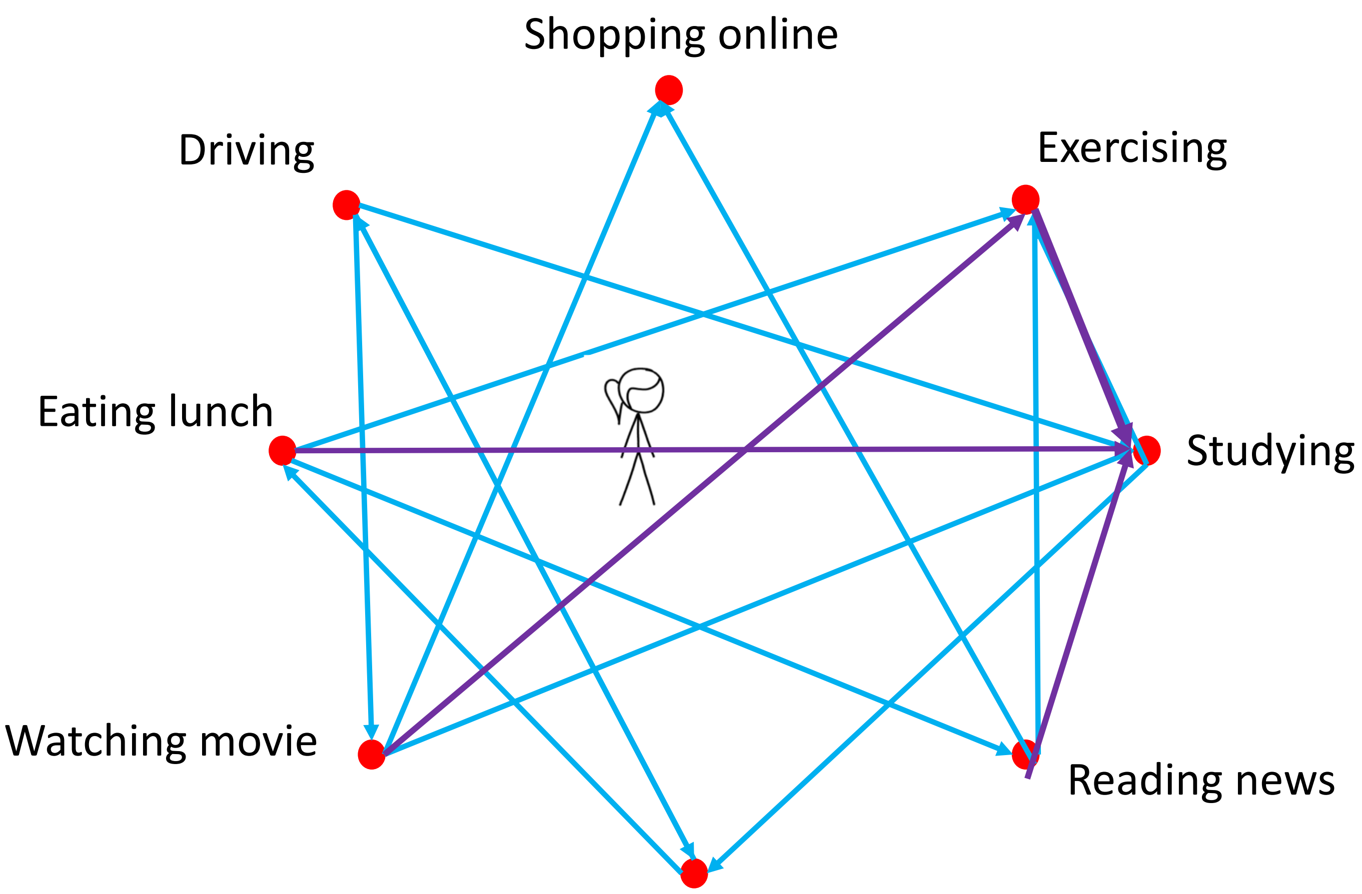
Multiple Agents

- Replace designer objective with summation over agents:

$$\text{profit}(S) := \sum_i \frac{\sum_{j \in \text{Agent}_i(S)} d_{ij} \cdot \frac{p_{ij}}{1-q_{ij}-y_{ij}}}{B_i + \sum_{l \in \text{Agent}_i(S)} z_{il}} - \sum_{j \in S} \text{cost}_j$$

- An exact polytime DP exists if #agents is constant.
  - Exponential in #agents
  - Also require potentials ϕ_i to be discretized by δ' with poly size.
- No FPTAS for 2 agents if ϕ_i not polynomial size.

Designer Competition



Agent's Life

What platforms should I build **to compete**?

Online firm

Competing firm

Multiple Platforms (Flower Setting)

- What if other competing designers have already built platforms?
  - Each platform affects only one state
  - At most one for each designer per state
- How does an agent behave?
- How should a designer optimally place platforms?

Multiple Platforms (Flower Setting)

- Agent’s algorithm is still greedy – but different potential function

- For platforms j, j' at the same state, define:

$$\rho(j, j') = \frac{z_{j'}\phi(j') - z_j\phi(j)}{z_{j'} - z_j}$$

- A “swap” potential: At state s, remove j and replace it with j'.

Multiple Platforms (Flower Setting)

- Is there an efficient designer algorithm?
- The multi-agent algorithm also (essentially) works in the multi-platform setting
    - Same discretization assumptions (potentials, denominator)
    - Exact algorithm
    - Polynomial time when #agents is constant
- Slight difference from old algorithm:
    - Modify the hash function: numerator and denominator of ψ instead

Summary

Recap

- Platform design: model economic activity of online firms
- General case of platform design is strongly NP complete.
- Tractable special case: the flower MDP
- Greedy agent algorithm
- Knapsack-style DP FPTAS for designer w/unbounded potentials
- Under polynomial, discretized potentials, exact DP for k agents (poly(n)$\cdot$ 2^k)
- Similar for multiple platforms
- **Many open directions!**

Future Work

Future Work

- Designer vs. designer
  - Complexity of pure Nash
  - Repeated game settings
- Privacy/fairness questions for agent
- Other classes of tractable MDPs?
- Results for generic classes of agent behavior?
- Many questions are problems of formulation