The Platform Design Problem

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

Columbia University

NetEcon 2021

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?

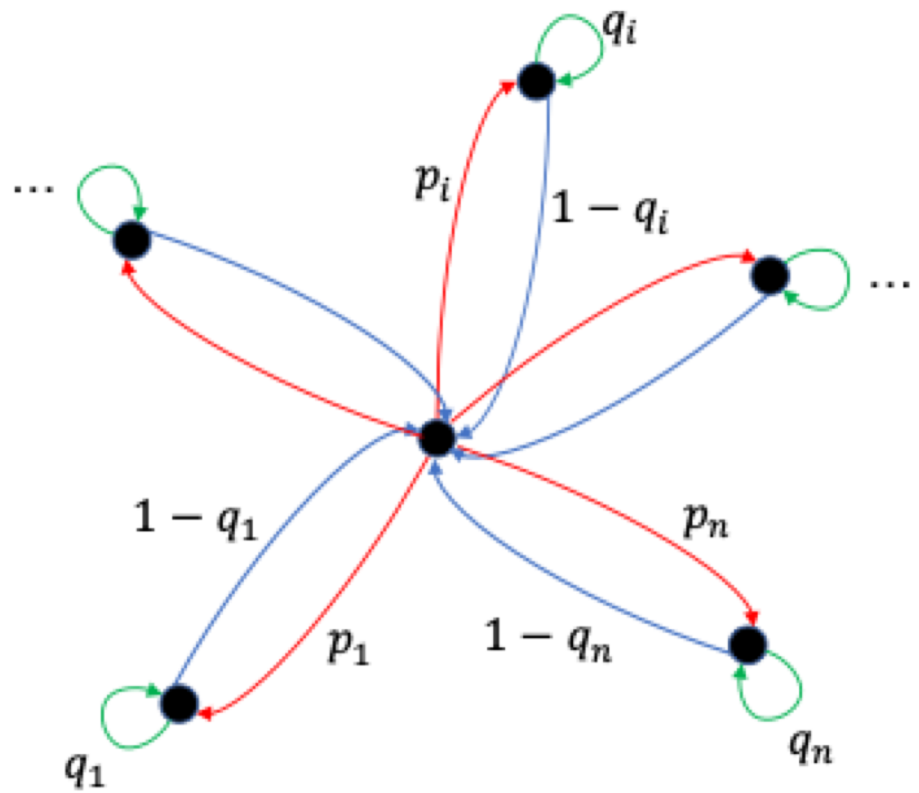
Computational Tractability I: General Case

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

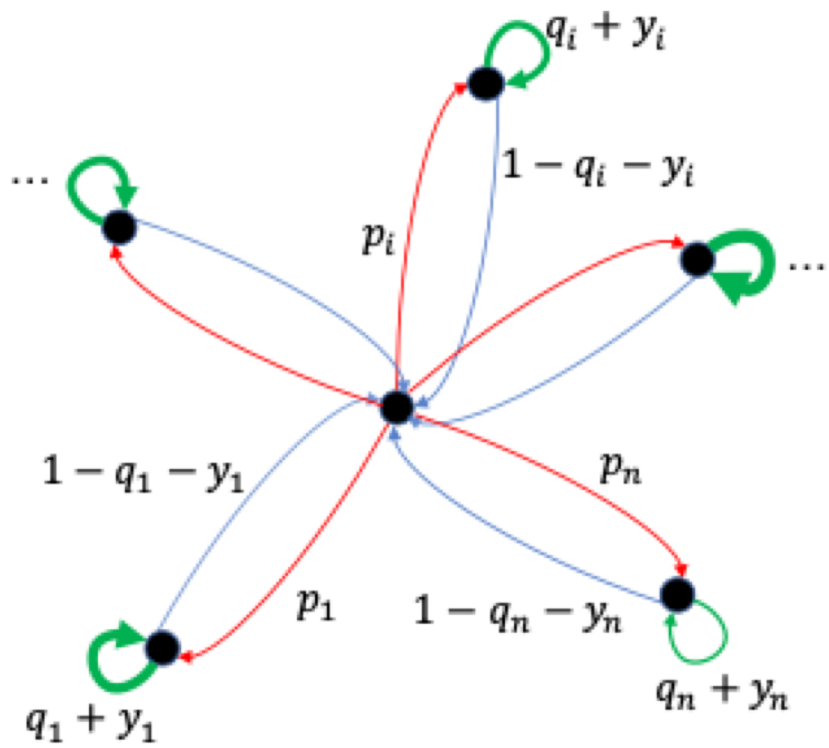
- Reduction from Set Cover
  - Designer builds platforms which each solve subset of Agent's problems.
  - Most cost-effective covering set is NP hard.

- 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.

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)

The Agent's Greedy Algorithm

- Sort states by potential function and add until utility = potential:

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):*

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

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 Designer's Dynamic Program

- Designer's profit function for set of platforms S:

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

- Assume z is discretized and costs are polynomially bounded
- Goal: (1 - ϵ) approximate algorithm in polynomial time.

The Designer's Dynamic Program

- Hash (total profit, revenue, revenue denominator) into a table
  - Scale the first two terms by ϵ * max profit/ num. states and round
  - Similar to standard Knapsack DP
- Store only platform sets that Agent accepts
  - Easy to simulate
- Update the platform set if revenue numerator is smaller
  - Smaller numerator + any successor set of states is feasible (Agent's behavior)
  - Profit is at least current profit minus ϵ * max profit/ num. states
  - Overall suboptimality is at most ϵ * max profit

Extensions

- Optimize rewards over many Agents
  - Similar DP exists, but exponential in # of Agent types
- Pre-Existing Designers
  - What if other Designers have already built platforms?
  - Similar DP exists

Future Work

- Designer vs. Designer
- We assumed everything is known to both sides
  - What about learning settings?
- Privacy/Fairness questions for Agent
- Many others…