The Evolution of Cooperation

Robert Axelrod, William D. Hamilton

Presented by Zarina Lin, Maggie Deng, Zena Hu

Motivation

- The rise of platforms such as Amazon has transformed pricing strategies with the use of pricing algorithms. These algorithms are **suspected** of unintentionally learning to collude, as collusion often leads to higher profits.
- Antitrust laws aim to prevent practices like tacit collusion, which harm consumers by increasing prices.

> The key concern is whether pricing algorithms **inherently foster** tacit collusion, raising prices for consumers.

- The model of repeated prisoner's dilemma in the Axelrod's tournament provides a good framework for simulation to address this concern.

Results of the Paper

Axelrod's tournament

- Iterated Prisoner's Dilemma

		Player B: C Cooperation	Player B: D Defection
Player A	C Cooperation	R=3 Reward for mutual cooperation	S=0 Sucker's payoff
	D Defection	T=5 Temptation to defect	P=1 Punishment for mutual defection

- Only consider interactions between two players
- Maximizing total payoff
- Game ends randomly
- Winner of the tournament: **TIT FOR TAT(TFT)**

What is TFT?

- Cooperate at first, then imitate last behavior of the other.
- Simplicity and Reciprocity
- Retaliation and Forgiveness
- Stability and Adaptability

Simulation – Set up

Assume

- market's demand function:

$$Q_i = 110 - P_i + \frac{1}{5} P_{-i}$$

  i stands for one of the firms, and $-i$ stands for all other firms
- marginal cost for each firm is identical, with $c = 10$
- profit function:

$$(P_i - c) \times Q_i$$

Using these assumptions, by maximizing profit, we found the Nash-Equilibrium price (deviating price) $P_{nash} = 100$, and the monopoly price (colluding price) $P_{mono} = 275$

Simulation – 5 Strategies

1. ALL-D: Play $P_{nash} = 100$ in each period.

```
def ALL_D():
    return p_nash
```

2. TFT: Play P_{mono} (cooperate) in period 1, then play the lowest price among P_{-i} in the previous round.

```
def TFT(p_list,i):
    return np.min(np.concatenate((p_list[0:i],p_list[i+1:])))
```

3. Max-Profit: Choose the price that maximizes the profit in response to the prices set by other firms in the previous round.

```
def max_profit(i, p_list):
    def profit_function(p_i):
        temp_p_list = p_list.copy()
        temp_p_list[i] = p_i
        return -profit(i, temp_p_list) #concave max= -convex min
    result = minimize_scalar(profit_function, bounds=(c, p_monopoly), method='bounded')
    return result.x
```

Simulation - 5 Strategies

4. Random: Randomly draw a price from the uniform distribution $[P_{nash}, P_{mono}] = [100, 275]$

```
def random():
    return np.random.uniform(p_nash, p_monopoly)
```

5. Avg: Set the price at the average price of the previous round.

```
def avg(p_list):
    return np.mean(p_list)
```

Simulation of playing different strategies – Test(code)

Test 1: 5 firms with 5 different strategies

```
def play_5_different(p_nash, p_monopoly, c):
    T = 10
    profit_matrix = np.zeros((5, T))
    price_matrix = np.zeros((5, T))
    accumulated_profit_matrix = np.zeros((5, T))

    p_list = [p_nash, np.random.uniform(p_nash, p_monopoly), p_monopoly, np.random.uniform(p_nash, p_monopoly), p_nash]

    for i in range(5):
        profit_i = profit(i, p_list)
        if profit_i > 0:
            profit_matrix[i, 0] = profit_i
        price_matrix[i, 0] = p_list[i]
        accumulated_profit_matrix[i, 0] = profit_matrix[i, 0]

    p_list = [p_nash, avg(p_list), TFT(p_list,3), random(), max_profit(4, p_list)]

    for k in range(1, T):
        p_list = [p_nash, avg(p_list), TFT(p_list,3), random(), max_profit(4, p_list)]
        for i in range(5):
            profit_i = profit(i, p_list)
            if profit_i > 0:
                profit_matrix[i, k] = profit_i
            price_matrix[i, k] = p_list[i]
            accumulated_profit_matrix[i, k] = accumulated_profit_matrix[i, k-1] + profit_matrix[i, k]

    return profit_matrix, price_matrix, accumulated_profit_matrix
```

Test 2: 5 firms with same strategy but different starting prices

```
p_list = [max_profit(4, p_list),max_profit(4, p_list), max_profit(4, p_list), max_profit(4, p_list), max_profit(4, p_list)]
```

```
p_init1=[p_monopoly, p_nash,p_monopoly,p_monopoly,p_nash]
p_init2=[p_monopoly, p_monopoly,p_monopoly,p_monopoly,p_monopoly]
```

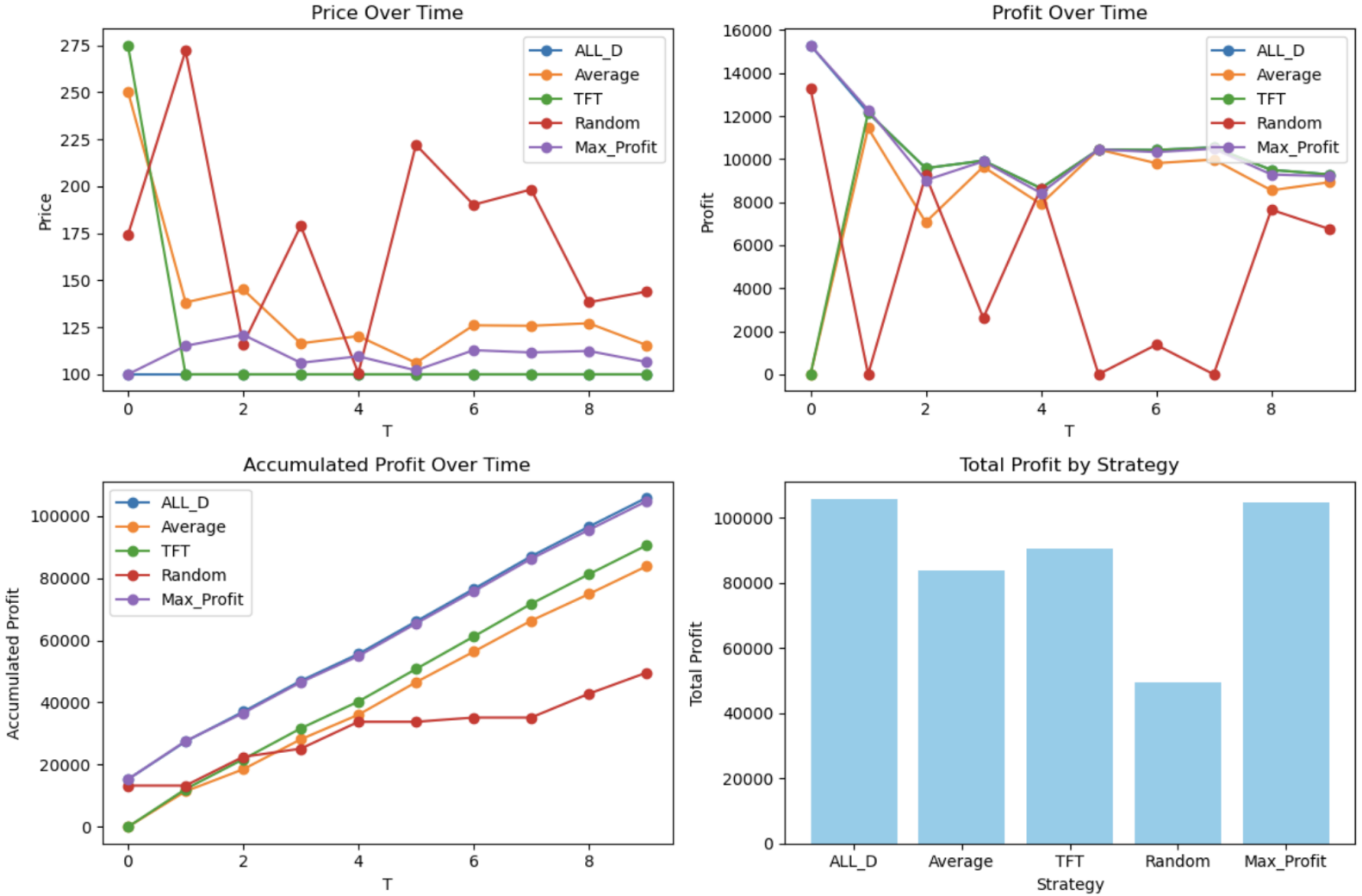
Test 3: 5 firms with All TFTs

```
p_list = [TFT(p_list,0),TFT(p_list,1), TFT(p_list,2), TFT(p_list,3), TFT(p_list,4)]
```

Test 4: 5 firms with 4 TFTs and 1 other start with cooperation

```
p_list = [max_profit(i, p_list),TFT(p_list,1), TFT(p_list,2), TFT(p_list,3), TFT(p_list,4)]
```

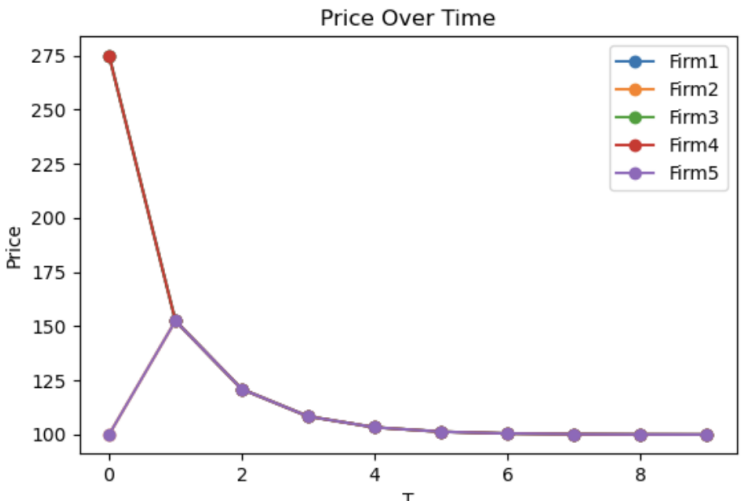
Test#1 - Comparing 5 Strategies

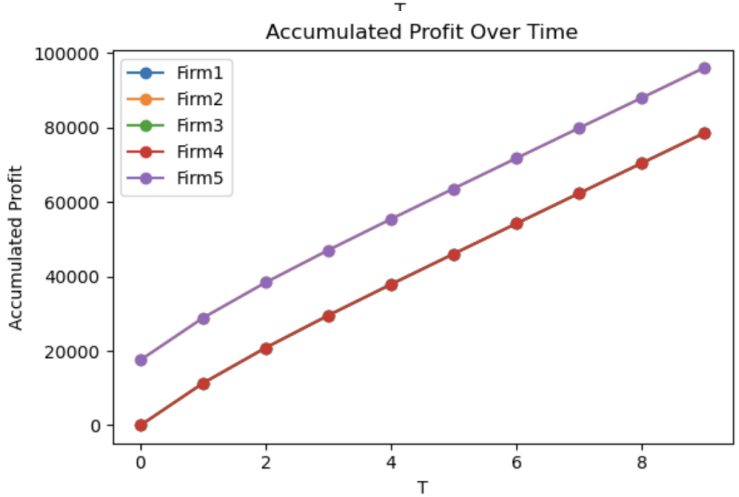


- Total profit: ALL-D > Max-Profit > TFT > Average > Random
- All-D is the best strategy

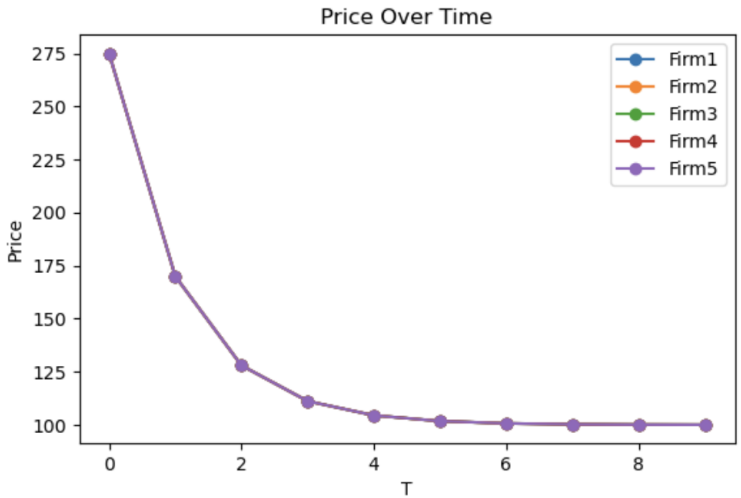
Test#2 - Max-Profit with different starting prices

Firm 1-4 starts with cooperating, firm 5 plays P_{nash}



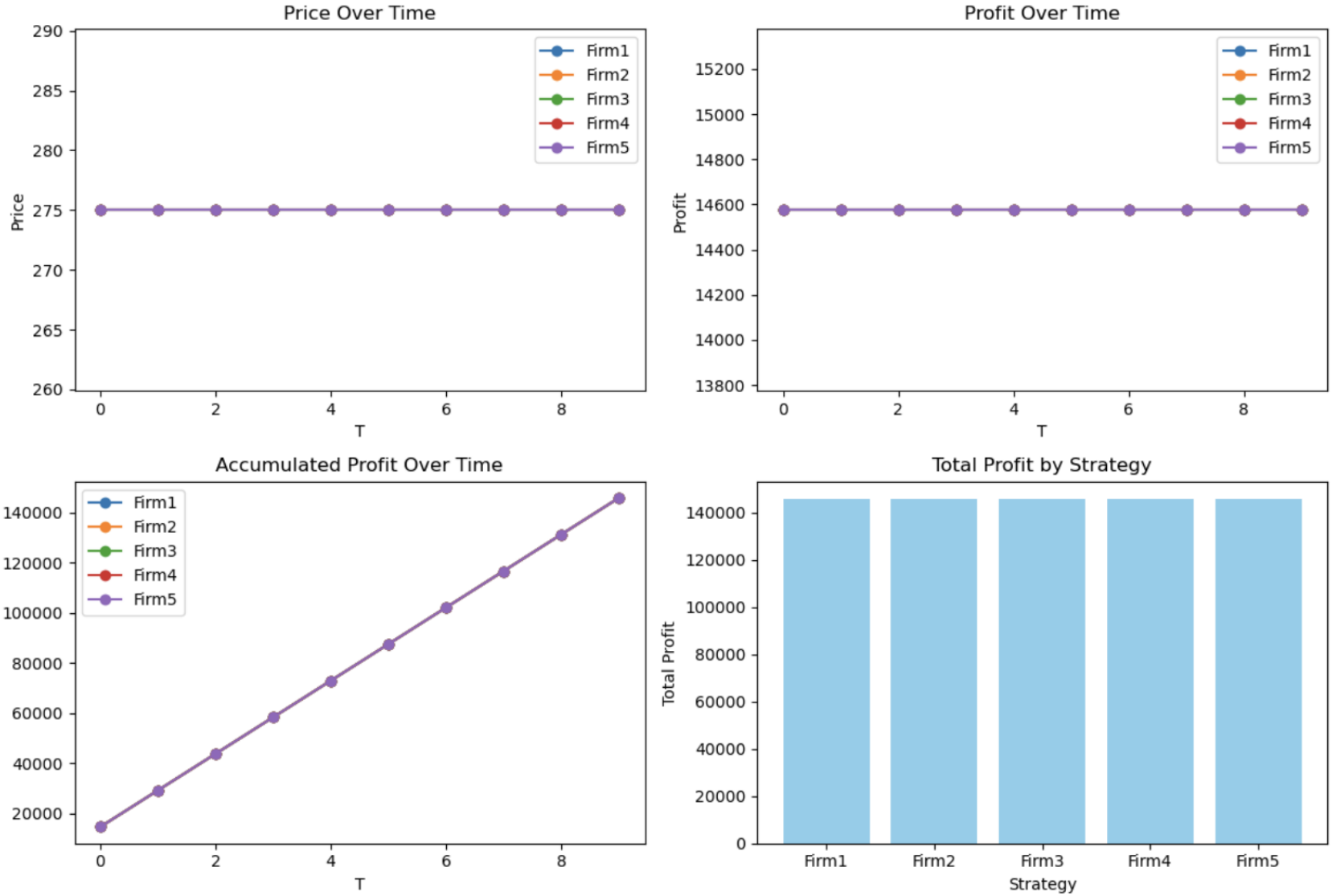


All 5 firms start with cooperating



- Both converge to P_{nash} at last, and firm 5 earns higher total profits
- Collusion cannot be sustained or arise by using Max-Profit algorithm

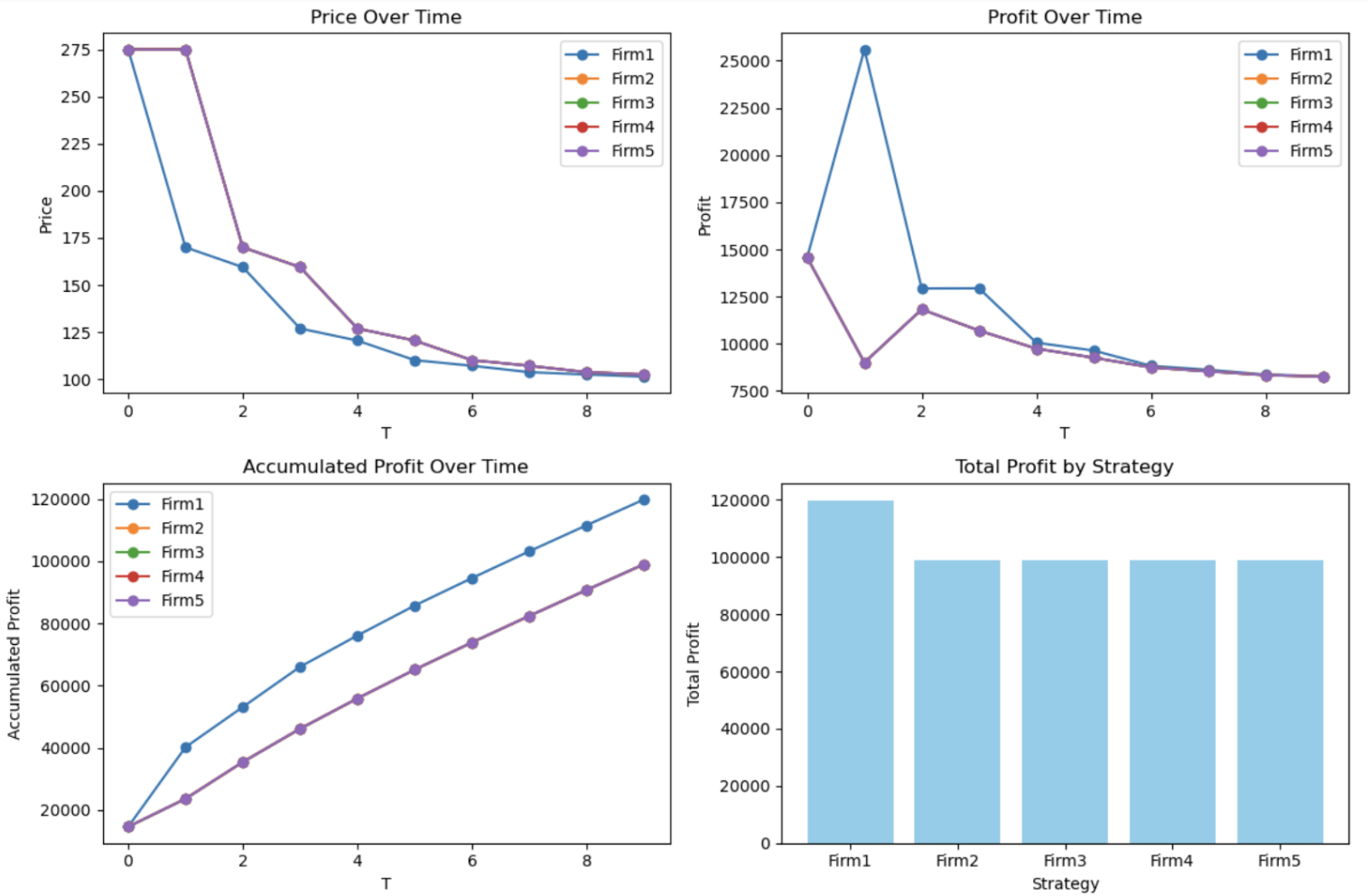
Test#3 - All play TFT



All firms playing TFT will make collusion stable and maximize the total profits, but lead to significantly higher prices.

Test#4-TFT v.s. Max-Profit

Firm 1 plays Max-Profit, and firms 2-5 play TFT. All start with playing P_{mono}



Collusion is unstable when at least 1 player does not use a colluding pricing algorithm, and that firm earns the highest total profit.

Conclusion

The simulation results suggest that:

- **Algorithms do not naturally tend to collude on their own:** Our results show that collusive algorithms, like the Tit-for-Tat (TFT) strategy, fail to achieve the highest profits unless all firms adopt the same approach. This indicates that spontaneous collusion based on past outcomes is unlikely.
- **Antitrust regulations should focus on deliberate collusion:** While it's important to regulate the intentional use of collusive algorithms, there's no strong evidence to suggest that algorithms inherently learn to collude. So antitrust agencies should avoid over regulation that may stifle innovation