Repeated n-player Coopetitive Polymatrix Games with Small Manipulation Cost

Zhijian Gao

August 2024

1 Introduction

Section 2 explains game setting for n players, section 3 provides two approaches, single-stroke and batch-coordination, to win, section 4 discuss what happens with stronger behavioural assumptions, and section 5 records progress.

2 Game Setting

$\Gamma = (\mathcal{N}, \mathcal{A}, u)$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of players, and $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$, where $\mathcal{A}_i$ is the action space of player i.

The strategy of player i is denoted by vector $x_i \in [0,1]^{|\mathcal{A}_i|}$ and $\Sigma_{j=1}^{|\mathcal{A}_i|} x_{i,j} = 1$ so that the entries of the strategies of each player are the corresponding probabilities that they play each action. Therefore, the probability of a combination of actions played each player, $a \in \mathcal{A}$ is $\mathbb{P}[a] = \Pi_{i=1}^{n} x_i(a_i)$, where $x_i(a_j)$ denotes the corresponding entry of action a_i in x_i. For example, if $n = 5$, and $a_1, \ldots, a_5$ are actions with indices $1, 3, 2, 4, 1$ in the corresponding action spaces of the 5 players, then $\mathbb{P}[a] = x_{1,1}x_{2,3}x_{3,2}x_{4,4}x_{5,1}$.

$\forall i \in \mathcal{N}, i \neq 1$ and $a = (a_1, \ldots, a_n) \in \mathcal{A}$, the utility function is

$$u_i(a) = \Sigma_{j \neq i} u_{i,j}(a) = \Sigma_{j \neq i} A^{(i,j)}(a_i, a_j)$$

where $A^{(i,j)}$ represents the utility matrix of player i when interacting with j. The utility function for player 1, the manipulator, is

$$u_1(a) = \Sigma_{j \neq 1}(A^{(1,j)}(a_1, a_j) - ||A^{(j,1)} - A_0^{(j,1)}||_\infty)$$

where $A^{(j,1)}$ represents the changed utility matrix, and $A_0^{(j,1)}$ are the original ones.

Hence, the expected utilities are

$$\mathbb{E}[u_1(a)] = \Sigma_{j=2}^{n}(x_1^T A^{(j,1)} x_j - ||A^{(j,1)} - A_0^{(j,1)}||_\infty)$$

and

$$\mathbb{E}[u_i(a)] = \Sigma_{j \neq i} x_i^T A^{(i,j)} x_j \forall i > 1$$

We repeat this game for T time steps. Other players do not have access to the manipulation matrices, but they have a history of their utilities and strategies at each action, and can choose their strategies based on the history.

The total utility of each player i is

$$U_i(x_1^{(t)}, \ldots, x_n^{(t)})_{t=1}^T = \frac{1}{T} \Sigma_{t=1}^T u_i(x_1^{(t)}, \ldots, x_n^{(t)})$$

where $x_i^{(t)}$ is the strategy of player i at time step t.

3 Problem Setting

We first assume that all other players are consistent.

Definition 1 (consistent agent): Suppose a^* is the unique best action for all rounds, and the number of rounds in which the agent chooses a^* in T rounds is T^*. If $\mathbb{P}[lim_{T \to \infty} \frac{T^*}{T} = 1] = 1$, then the agent is a consistent agent. If such an action doesn't exist, then we don't make any assumption on the behaviour of this agent.

The goal is to make sure that

- player 1 wins
- the manipulation cost is minimised

3.1 First Approach

We approach this by manipulating payoff matrices to ensure that each player has a unique best action $a_{l_j^*} \in \mathcal{A}_j$ that is the optimal choice for every round, and player 1 has a unique action such that if it plays this action, it will win in the long run. i.e. we must find matrices $A^{(j,1)}$ for every $j \neq 1$ such that

- $u_j(a_{a_1 = l_1^*, a_j = l_j^*}) > u_j(a) \forall a$, where $a_{a_1 = l_1^*, a_j = l_j^*}$ is the vector of combination of actions formed by replacing the $1st$ and jth entries by the unique best actions l_1^* and l_j^* respectively
- $u_1(e_{l_1^*}, \ldots, e_{l_n^*})_{t=1}^T \geq max_{j \neq 1} u_j(e_{l_1^*}, \ldots, e_{l_n^*})_{t=1}^T$
- $\Sigma_{j \neq 1} ||A^{(j,1)} - A_0^{(j,1)}||_\infty$ is minimised

3.1.1 Breaking Down

This means that we must choose $A^{(j,1)}$ such that for some $v_j \in \mathbb{R}^{|M_j|}$, where $M_j = \{(l_k)_{k\neq 1, k\neq j} : a_{l_k} \in \mathcal{A}_k\}$ is the set of all possible combinations of actions with the actions of player 1 and player j fixed to their unique strictly dominant action, we have

$$e_{l_1^*}^T A^{(j,1)} e_{l_j} + \Sigma_{k\neq 1, k\neq j} e_{l_k}^T A^{(j,k)} e_{l_j} = v_{j,m} \forall m \in [|M_j|] \text{ and } l_j = l_j^*$$

$$e_{l_1^*}^T A^{(j,1)} e_{l_j} + \Sigma_{k\neq 1, k\neq j} e_{l_k}^T A^{(j,k)} e_{l_j} < v_{j,m} \forall m \in [|M_j|] \text{ and } l_j \neq l_j^* \quad (1)$$

where l_j^* is the unique best action in all rounds for player j against the rest, and $m \in [|M_j|]$ is the index of the mth largest vector $(l_k) \in M_j$ when they are interpreted as $n-2$-digit integers. For example, if there are 5 players, each player have 6 actions and we let $j = 3$, then the combination of strategies (1, 2, 6) will correspond to an m value of 12.

Other than this, we must also make sure that

$$\Sigma_{j\neq 1}(e_{l_1^*} A^{(1,j)} e_{l_j^*} - ||A^{(j,1)} - A_0^{(j,1)}||_\infty) \geq max_{j\neq 1} \Sigma_{k\neq j} e_{l_j^*} A^{(j,k)} e_{l_k^*} \quad (2)$$

and minimise

$$\Sigma_{j\neq 1} ||A^{(j,1)} - A_0^{(j,1)}||_\infty \quad (3)$$

3.1.2 Solution

Definition 2 (Dominance Solvable Optimal Policy, DSOP): A dominance solvable policy is a tuple $\rho = (e_{l_1^*}, A^{(2,1)}, \ldots, A^{(n,1)})$ that satisfies (1), (2) and minimises (3).

Proposition 1: If player 1 plays the DSOP against consistent agents and the number of time steps is infinite, then it will win, i.e.

$$\mathbb{P}[U_1(x_1^{(t),\ldots,x_n(t)})_{t=1}^\infty > max_{j\neq 1} U_j(x_1^{(t),\ldots,x_n(t)})_{t=1}^\infty | \text{player 1 plays DSOP}] = 1$$

Proof: Trivial. When DSOP is played by player 1, eventually other players will learn their strictly dominant strategy and player 1 will start to have more time-step-wise utility, and eventually will win after several time steps.

Theorem 1: DSOP cannot be computed in polynomial time wrt the number of players through linear programming, but can be computed in polynomial time wrt the number of actions.
Proof: Finding a tuple $(e_{l_1^*}, A^{(2,1)}, \ldots, A^{(n,1)})$ that satisfies (1), (2) and (3) is the same as a constrained optimisation problem, which can in fact be transferred to an LP problem:

$$min \Sigma_{j\neq 1} \alpha_j \text{ subject to}$$

$$A_{p,q}^{(j,1)} - A_{0_{p,q}}^{(j,1)} \leq \alpha_{j_{p,q}} \quad \forall p \in |\mathcal{A}_1|, q \in |\mathcal{A}_j| \forall j \neq 1 \quad (C1)$$

$$A^{(j,1)}_{0_{p,q}} - A^{(j,1)}_{p,q} \leq \alpha_{j_{p,q}} \quad \forall p \in |\mathcal{A}_1|, q \in |\mathcal{A}_j| \forall j \neq 1 \quad (C2)$$

$$\Sigma_q \alpha_{j_{p,q}} \leq \alpha_j \quad \forall j \neq 1, \text{ and } p \in |\mathcal{A}_j| \quad (C3)$$

$$\Sigma_{j \neq 1}(A^{(1,j)}_{l_1^*, l_j^*} - \alpha_j) \geq \Sigma_{k \neq j} A^{(j,k)}_{l_j^*, l_k^*} \quad \forall j \neq 1 \quad (C4)$$

$$A^{(j,1)}_{l_1^*, l_j} + \Sigma_{k \neq 1, k \neq j} A^{(j,k)}_{l_k, l_j} < A^{(j,1)}_{l_1^*, l_j^*} + \Sigma_{k \neq 1, k \neq j} A^{(j,k)}_{l_k, l_j^*} \quad \forall j \neq 1, l_j \neq l_j^*, (l_k)_{k \neq 1, k \neq j} \in \mathcal{M}_j \quad (C5)$$

$C1$ to $C3$ makes sure that we don't have to deal with absolute values. Each $\alpha_{j_{p,q}}$ represents $|A^{(j,1)}_{p,q} - A^{(j,1)}_{0_{p,q}}|$, and $\alpha_j = ||A^{(j,1)} - A^{(j,1)}_0||_\infty$. After that, $C4$ is the same as (2), and $C5$ is the same as (1).
Since this is an LP problem, we can solve it using Karmarkar's algorithm, which solves LP problems in $O(r^{1.5}s^2L)$, where r is the number of constraints in inequalities, s is the number of variables and L is the precision in bits. We now analyse them one by one.

- #inequalities in $C1$ =#inequalities in $C2 = \Sigma_{j=2}^{n} |\mathcal{A}_1||\mathcal{A}_j|$
- #inequalities in $C3 = (n-1)|\mathcal{A}_1|$
- #inequalities in $C4 = n - 1$
- #inequalities in $C5 = \Sigma_{j \neq 1}(|\mathcal{A}_j| - 1)|\mathcal{M}_j| = \Sigma_{j \neq 1}(|\mathcal{A}_j| - 1)\Pi_{k \neq 1, k \neq j}|\mathcal{A}_k|$

Let $\beta = max_j|\mathcal{A}_j|$. Then, $r \approx 2(n-1)\beta^2 + (n-1)\beta + n - 1 + (n-1)(\beta - 1)\beta^{n-2} = (n-1)(2\beta^2 + \beta^{n-2}(\beta - 1) + \beta + 1), s$ =#entries in all matrices $A^{(j,1)}$+#α_js+#$\alpha_{j_{p,q}}$s$\approx (n-1)\beta^2 + (n-1)(1+\beta^2) = (n-1)(2\beta^2+1)$. Therefore, the running time would be

$$O((n-1)^{1.5}(2\beta^2 + \beta^{n-2}(\beta-1) + \beta + 1)^{1.5}(n-1)^2(2\beta^2+1)^2 L) = O(\beta^{1.5n+2.5})$$

QED.

Conjecture 1: The problem is NP hard wrt the number of players n.
Proof/Disproof:

3.2 Second Approach

Intuitively, if we focus on beating each player one by one and in the mean time ignore the others, player 1 should be able to win with a even lower total manipulation cost.

3.2.1 Reforming the problem

We'll separate the game into $n-1$ different periods of time steps follow one DSOP for each period, only this time, we focus on beating one player at each period. Therefore, for each period $T_c = [\frac{(c-2)T}{n-1}, \frac{(c-1)T}{n-1}], c = 2, \ldots, n$, we find matrices $A^{(j,1)}_c$ such that $\forall j \neq 1$, we need to find $v_j^(c) \in \mathbb{R}^{|\mathcal{M}_j|}$ such that

$$e^T_{l_1^{*(c)}} A^{(j,1)} e_{l_j} + \Sigma_{k \neq 1, k \neq j} e^T_{l_k} A^{(j,k)} e_{l_j} = v_{j,m} \forall m \in [|M_j|] \text{ and } l_j = l_j^{*(c)}$$

$$e^T_{l_1^{*(c)}} A^{(j,1)} e_{l_j} + \Sigma_{k\neq 1, k\neq j} e^T_{l_k} A^{(j,k)} e_{l_j} < v_{j,m} \forall m \in [|M_j|] \text{ and } l_j \neq l_j^{*(c)} \quad (4)$$

Then, we need to make sure that

$$\mathbb{E}[\Sigma^n_{c=2}\Sigma_{t\in T_c} u_1(e_{l_1^{*(c)}}, \ldots, e_{l_n^{*(c)}})] \geq max_j \mathbb{E}[\Sigma^n_{c=2}\Sigma_{t\in T_c} u_j(e_{l_1^{*(c)}}, \ldots, e_{l_n^{*(c)}})]$$

so that even though player 1 may be temporarily loosing to some players in some of the periods, eventually it will win.

Other than this, we want to minimise

$$\Sigma^n_{c=2}\Sigma_{j\neq 1} ||A^{(j,1)}_c - A^{(j,1)}_0||_\infty$$

3.2.2 Solution

Definition 3 (dominance solvable batch coordination optimal policies, DSBCOP): This is a policy

$$\rho_t = \begin{cases} (e_{l_1^{*(2)}}, A^{(2,1)}_2, \ldots, A^{(n,1)}_2) & \text{for } 1 < t \leq \frac{T}{n-1} \\ (e_{l_1^{*(3)}}, A^{(2,1)}_3, \ldots, A^{(n,1)}_3) & \text{for } \frac{T}{n-1} < t \leq \frac{2T}{n-1} \\ \ldots \\ (e_{l_1^{*(n)}}, A^{(2,1)}_n, \ldots, A^{(n,1)}_n) & \text{for } \frac{(n-2)T}{n-1} < t \leq T \end{cases}$$

that satisfies (4), (5) and minimises (6).

Proposition 2: If player 1 plays DSBCOP against consistent players, it will win.
Proof: This is guaranteed by construction of the policy.

Theorem 2: DSBCOP cannot be computed in polynomial time wrt the number of players through linear programming, but can be computed in polynomial time wrt the number of actions.
Proof: We can turn this into a linear programming problem in the same manner as DSOP:

$$min \Sigma^n_{c=2}\Sigma_{j\neq 1} \alpha^{(c)}_j \text{ subject to}$$

$$A^{(j,1)}_{c_{p,q}} - A^{(j,1)}_{0_{p,q}} \leq \alpha^{(c)}_{j_{p,q}} \quad \forall c \geq 2, j \neq 1, p \in |\mathcal{A}_1| \text{ and } q \in |\mathcal{A}_j| \quad (C6)$$

$$A^{(j,1)}_{0_{p,q}} - A^{(j,1)}_{c_{p,q}} \leq \alpha^{(c)}_{j_{p,q}} \quad \forall c \geq 2, j \neq 1, p \in |\mathcal{A}_1| \text{ and } q \in |\mathcal{A}_j| \quad (C7)$$

$$\Sigma_q \alpha^{(c)}_{j_{p,q}} \leq \alpha^{(c)}_j \quad \forall c \geq 2, j \neq 1, p \in |\mathcal{A}_1| \quad (C8)$$

$$\Sigma^n_{c=2}\Sigma_{j\neq 1}(A^{(1,j)}_{c_{l_1^{*(c)}, l_j^{*(c)}}} - \alpha^{(c)}_j) \geq \Sigma^n_{c=2}\Sigma_{k\neq j} A^{(j,k)}_{c_{l_j^{*(c)}, l_k^{*(c)}}} \quad \forall j \neq 1 \quad (C9)$$

$$A^{(j,1)}_{c_{l_1^{*(c)}, l_j}} + \Sigma_{k\neq 1, k\neq j} A^{(j,k)}_{c_{l_k, l_j}} < A^{(j,1)}_{c_{l_1^{*(c)}, l_j^{*(c)}}} + \Sigma_{k\neq 1, k\neq j} A^{(j,k)}_{c_{l_k, l_j^{*(c)}}} \forall j \neq 1, l_j \neq l_j^*, (l_k)_{k\neq 1, k\neq j} \in \mathcal{M}_j, c = 2, \ldots, n (C10)$$

Let the number of constrains and number of variables be r' and s' respectively in this case. Then, we have $r' = 2(n-1)^2\beta^2+(n-1)^2\beta+n-1+(n-1)^2(\beta-1)\beta^{n-2}$ and $s' = (n-1)\beta^2 + (n-1)^2(1+\beta^2)$. Therefore, the running time is

$$O((n-1)^{1.5}(2(n-1)\beta^2+(n-1)\beta+1+(n-1)(\beta-1)\beta^{n-2})^{1.5}(n-1)^2(\beta^2+(n-1)(1+\beta^2))^2L)$$

$$= O(\beta^{1.5n+2.5})$$

QED.

Conjecture 2: DSECOP can be more optimal than DSOP.
Proof/Disproof: Proof in progress. I'm trying to formulate an example because I think this might be true.

4 Stronger Behavioural Assumptions

In this section we alter and examine the policies on players with stronger behavioural assumptions.

4.1 Persistent Player

Definition 4 (persistent agent): The agent who is consistent on the eventually best action once it is revealed.

4.2 No-Regret Player

Definition 5 (regret): The regret of any sequence of strategies $(y_1, \ldots, y_T)$ chosen by P2 wrt a fixed strategy y is

$$R_{T,y} = \Sigma_{t=1}^{T} x_t A_t^{(2,1)} y_t + y_t^T A_0^{(2,3)} z_t - \Sigma_{t=1}^{T} x_t^T A_t^{(2,1)} y + y^T A_0^{(2,3)} z_t$$

i.e., the difference between the payoff accumulated by $(y_1, \ldots, y_T)$ and the payoff accumulated by $(y, \ldots, y)$.

Definition 6 (no-regret agent): An agent whose choices of strategies satisfy

$$lim_{T\to\infty} max_y \frac{R_{T,y}}{T} = 0$$

Proposition 3: No-regret agents are persistent agents, and persistent agents are consistent agents.
Proof: Trivial. Will be detailed after other problems are solved.

5 Periodic Summary

19.08.2024: Proof of Theorem 1 and Theorem 2 shows that no matter conjecture 2 is right or not, it's possible for player 1 to find a winning policy that

allows it to minimise the manipulation cost within polynomial time wrt number of actions. The task for next period would be to prove or disprove conjecture 1 (DSOP is NP hard wrt n) and 2 (executing DSBCOP can win with lower manipulation cost). Intuitively Conjecture 2 is true, planning to form proof by finding supporting example through Python.

21.08.2024: A problem in representing the infinity norm for LP was detected, representation had been changed and it didn't cause problem for the overall running times. Problem fixed. Started to try to prove conjecture 1 by using 3-SAT problem as the source problem and consider the situation in which $|\mathcal{A}_j| = 2 \quad \forall j$

23.08.2024: 3-SAT won't work. Now Trying TSP, building polynomial-time reduction by using the payoff matrices as buildings and infinity norm as distances between buildings. Trying to figure out how to represent order of buildings.

26.08.2024: TSP won't work, the permutation of the buildings in the output cannot correspond to any aspect of our problem.