Large Markov Game solving with aggregators

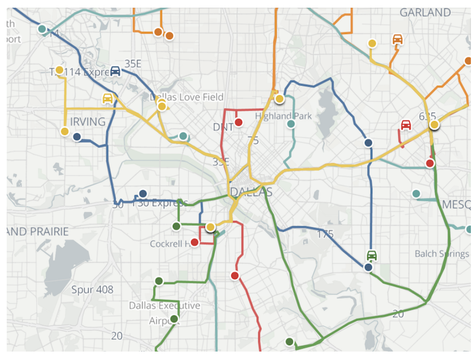
Yuanchen Tang [1] Jingwu Tang [2] Zhiwei Steven Wu [2]

[1]Peking University [2]Carnegie Mellon University

Large game with aggregators

We study Markov game involving ***a large number of participants*** where a multi-dimensional aggregator exist in every state(or aggregative game). We aim to solve a Nash Equilibrium(NE) of this class of game.

Aggregative game is a broad model for large games where players' actions combine into an aggregator vector, which consists of linear functions of these actions. Each player's utility is a potentially non-linear function of both this aggregator and their own action.



E.g., Routing Game: *"Every driver in the city: I want to pick a favourite route"*
An aggregator represents the level of congestion in each region or on each road.

Large aggregative game means that any player's unilateral change in action can have at most a bounded influence on the utility of any other player.

Key Observation: The computation can be reduced to one-stage equilibrium computation assuming the bounded influence of transition function on single deviation.

Background

ϵ-Nash Equilibrium A collective strategy profile is called an ϵ-Nash Equilibrium if every participant cannot obtain an increase greater than ϵ in utility by changing strategy alone.

Markov Game The multi-agent version of Markov Decision Process(MDP) Here we will try to solve *finite-horizon general-sum* Markov Games with aggregators where policies are independent every timestep.

In general, approximate Nash equilibrium is PPAD-hard in both normal-form and Markov Games. Hence, people typically focus on polynomially approximable relaxed notions of NE where agents are correlated. However in aggregative games, we can address this challenge by leveraging state-wise aggregators.

Notation

$MG(H, \mathcal{S}, \mathcal{A}, \mathcal{T}, \{r_i\}_{i=1}^m)$

H: horizon. m: number of agents.

$\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_m$: the joint action space for all agents.

$S_h(\vec{a}) \in [-W, W]^d$: aggregator.

$r_{i,h}(s, \vec{a}), r_{i,h}(s, a_i, S) \in [0, 1]$: reward function without/with aggregator.

$P_h(s, \vec{a}), P_h(s, S) \in \Delta(\mathcal{S})$: transition function without/with aggregator.

$V_{i,h}(s)$: state value function.

$Q_{i,h}(s, \vec{a}), Q_{i,h}(s, a_i, S)$: state-action value function without/with aggregator.

Two properties of aggregator:

1. $||S(a_i, a_{-i}) - S(a_i', a_{-i})||_\infty \leq \delta$: Bounded influence
2. $|r_i(s, a_i, S) - r_i(s, a_i, S')| \leq \gamma_r ||S - S'||_\infty$ Lipschitz
3. $|P(s, S) - P(s, S')|_{TVD} \leq \gamma_P ||S - S'||_\infty$ Lipschitz

[1] Rachel Cummings et al. "Privacy and truthful equilibrium selection for aggregative games". In: *Web and Internet Economics: 11th International Conference*. Springer. 2015, pp. 286–299.

One-stage game: Best response to aggregators

Thanks to the properties of aggregators, an ϵ-Nash equilibrium in a normal-form aggregative game exists if and only if each agent has approximately best-responded to the current strategy profile's aggregator. This significantly simplifies the constraints.

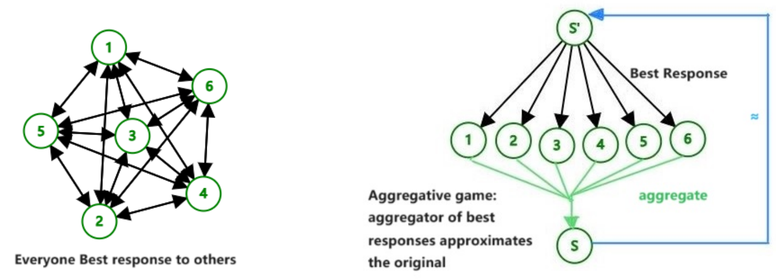


Figure 1. An intuitive representation of constraints of Nash Equilibrium. We see original game NE requires complicated constraints(Left), while aggregative game NE shows a much easier constraint structure(Right)

For every fixed aggregator, linear programming is sufficient to verify existence of equilibrium. It suffice to search through a discretized grid of all possible aggregators.[1]

Markov game: Formalizing Aggregate-VI

The goal is to find a policy $\{\pi_i\}_{i\in[n]}$ such that:

$$\max_{i\in[n]}(\sup_{\pi_i'} V_{1,i}^{\pi_i', \pi_{-i}} - V_{1,i}^{\pi})(s_1) \leq \epsilon.$$

We propose trying to compute suffix equilibrium inductively. Algorithmically, game-solving for a large Markov Game boils down to this simple procedure:

Aggregator Value Iteration(`Aggregate-VI`)

For timestep $h = H, H-1, ..., 1$:

1. Estimate the state-action value by $Q_{i,h}(s, a_i, S) \leftarrow r_{i,h}(s, a_i, S) + \sum_{s'} P(s, S, s') V_{i,h+1}(s')$ for all s, i, a_i, S, where $S \in \{S_k, k \in Z \cap [-W/\alpha, W/\alpha]\} \cup \{S_h(s, \vec{a}_i)\}$
2. Solve a linear programming(LP) problem on estimated Q-function to get the policy on the current timestep from $\pi_h(\cdot|s) \leftarrow LP(\{Q_{i,h}(s, a_i, S_k)\}_{i,k,a_i})$ for all state s.
3. Estimate state value by $V_{i,h}(s) \leftarrow \mathbf{D}_{\pi_h} Q_{i,h}(s, \cdot)$ for all s, i.

Output $\{\pi_h\}_{h\in[H]}$

Under a probabilistic assumption on transition function that reflects continuity on aggregator, we provide a tight bound on the approximation factor.

Theorem: For discretizing interval length α, our algorithm obtains an $(O(\alpha\gamma_r + \delta\gamma_r)H + O(\delta\gamma_P)H^2)$-Nash Equilibrium.

Future Work

1. Design social welfare maximizing algorithms.
2. Construct sample-efficient learning algorithms.
3. Test empirical performance of these algorithms on real-world data.

Acknowledgement

I would like to express my heartfelt thanks to my advisor Professor Steven Wu, my mentor Jingwu Tang, and the members of the Software and Societal Department for their invaluable help and support. I'm also grateful to the CMU-PKU CS summer research internship program for providing me with this incredible experience. A special thanks to my friends for always being there for me!