Mechanisms for the evolution of prosociality

Graser, C.J.

Publication date
2024

Chapter 4

Best responder algorithms for mixed populations and for endogenous separation

4.1 Abstract

García and van Veelen (2018) developed an algorithm to identify best responses against pure strategies in repeated games. Here, we extend this algorithm to be able to identify optimal play in mixed populations, where players face opponents playing different strategies at different frequencies. Moreover, we present an adaptation of this algorithm with which we can identify optimal play for settings in which the duration of the repeated game is not exogenous, but in which players can terminate the interaction with their current partner and re-match with a randomly drawn new partner.

4.2 Setup

We consider a repeated two-player game with finite action space A at each stage-game. A pure strategy, here, is a deterministic map from the space of possible histories of previous play between the two players to the set of stage game actions. Using the notation in García and van Veelen (2018), we denote the histories at the t'th round of play as $h_t = ((a_{1,1}, a_{1,2}), \ldots, (a_{t-1,1}, a_{t-1,2}))$, where subscripts of the actions $a \in A$ specify time and player respectively. Moreover, the set of histories H is simply the union of the sets of possible histories H_t at each round t: $H = \cup_{t=1}^{\infty} H_t$.

García and van Veelen (2016) show that pure strategies in the repeated prisoner's dilemma (or any other repeated game) can be represented, or at least be approximated arbitrarily well, by finite state automata. A finite state automaton in our setting refers to collection of n states, endowed with two maps. The first map, λ, associates a stage-game action $\lambda(i) \in A$ with every state i. The second map, $\mu : \{1, \ldots, n\} \times A^2 \mapsto \{1, \ldots, n\}$ specifies how the automaton transitions between states depending on the actions (a_1, a_2) that are played.

One advantage of representing strategies as finite state automata is that it simplifies the search for a best response to a given strategy. At any given round t_1, a player's expected future payoff for playing a sequence of own actions $\{a_t\}_{t_1}^{\infty}$ is determined fully by the state that the opponent automaton is in at round t_1. Moreover, if in a given state i of the opponent it is optimal to play a sequence starting with some action $a_{t_1}^*(i)$ at t_1, then this action $a_{t_1}^*(i)$ will also be optimal at any future round at which the opponent is in state i. Therefore it is possible to describe a strategy that plays a best response against a finite state automaton with n states, as a finite state automaton that also has with n states. This best-responding automaton simply follows all state transitions of its opponent, and at every state of its opponent plays the action $a^*(i)$.

As a consequence, rather than having to consider all possible automata to identify a best response, it suffices to consider only automata with n states. However, since the set of n-state automata might, nevertheless, be so large that considering all n-state automata is computationally expensive, García and van Veelen (2018) have developed an algorithm that more efficiently traverses the set of n-state automata to find an a best response. This algorithm is described below. For a proof that this algorithm converges to a best response see García and van Veelen (2018).

Algorithm 1: Best Responder (García and van Veelen, 2018)
Input: (1) An automaton $M = \{\{1, ..., n_M\}, \lambda_M, \mu_M\}$, (2) a payoff mapping for the stage game $\pi : A^2 \mapsto \mathbb{R}_+$[1], and (3) a discount factor for the repeated game δ, which might reflect a continuation probability.
Procedure: (1) Start with an n-element vector of zeros $V_0 = (0, \ldots, 0)$.
(2) In the i'th round of the algorithm, for every state $k \in \{1, \ldots, n_M\}$, solve

$$V_i(k) = \max_{a \in A} \left(\pi(a, \lambda(k)) + \delta V_{i-1}(\mu(k, \lambda(k), a)\right)$$

(3) Repeat step (2) until a round passes at which $\forall k : V_i(k) = V_{i-1}(k)$.
Output: A set of optimal actions a_k^* in every state k, and a vector $V^* = V_i$ of expected future payoffs for the best responding automaton, for every state k that the opponent automaton can be in.

Of course, best responses need not be unique. If in the final iteration of Alrogithm 1, the opponent automaton M has a state k for which there are multiple actions $\{a_k, a'_k, \ldots\} \in A$ that maximize the expression for the k'th element in step (2) of the procedure, then we can, for instance, change the behaviour of the output in state k to then any mapping from the set of histories that put M in state k, to the actions $\{a_k, a'_k, \ldots\}$ that maximize the expression in step (2), and this changed strategy still constitutes a best response. The best responses identified in

[1]Note that requiring that π maps to $\mathbb{R}_+$ is no restriction of generality for games with finite action space A, because any $\pi' : A^2 \mapsto \mathbb{R}$ can be transformed to $\pi : A^2 \mapsto \mathbb{R}_+$ by adding a constant to all payoffs, which, in turn, has no influence on what actions maximize payoffs.

Algorithm 1, have the feature that they play a unique action for every state that M can be in. Note, however, that this feature does not imply that the best-responding automaton that Algorithm 1 finds is minimal – it is certainly possible that the behaviour of this automaton can be represented by an automaton with fewer states.

While Algorithm 1 can be used to find a best response against a given automaton – or to identify optimal play in population of individuals that all play the same strategy –, it is not suited to identify best responses in mixed populations. Here, we present an extension to Algorithm 1 to mixed populations.

4.3 Best responder algorithm in mixed populations

We first present our approach for the case of a mixed population with two strategies represented as finite state automata M_1 and M_2, and afterwards, we discuss how this approach extends to higher numbers of strategies. We assume the player we consider is randomly matched with others in the mixed population with probabilities $(p_1, 1-p_1)$. These probabilities need not correspond to the actual frequencies of M_1 and M_2 in the population. Our algorithm, therefore, works not only for well-mixed populations in which automata are with uniform probability randomly matched with other automata, but for arbitrary matching procedures.

At the beginning of the interaction, the player we consider does not know whether their opponent is of type M_1 or M_2. However, this uncertainty can resolve over time, if given a history h_t is reached which could have only been generated by playing against one but not the other automaton. As a first step, we, thus, require a procedure that identifies the possible histories that result in such a revelation of the opponent's type. We refer to actions that are played before the opponent's type is revealed as blind shots. Algorithm 2 produces a tree that represents all possible non-repeating sequences of such blind shots.

Algorithm 2: Blind Shot Tree
Input: $M_1 = \{\{1, ..., n_{M_1}\}, \lambda_{M_1}, \mu_{M_1}\}$, $M_2 = \{\{1, ..., n_{M_2}\}, \lambda_{M_2}, \mu_{M_2}\}$
Procedure (Sketch): We generate a rooted tree with branching factor $|A|$ to identify and represent the possible histories that can occur until the opponent automaton can be identified as either M_1 or M_2. We associate an action $a \in A$ with each edge of this tree, and with each interior node we associate a triplet $(h_t, i_1(h_t), i_2(h_t))$ of a history h_t and the states i_1 and i_2 that M_1 and M_2 are in after h_t.
The root is initialized with the states $i_1 = i_2 = 1$ and the empty history $h_1 = ()$. The algorithm iteratively considers leaf nodes of the tree and expands the tree by adding branches to these leaf nodes. The histories associated with the $|A|$ nodes that branch off from a mother node with history h_t are those that extended h_t with a different $a \in A$, paired with the action that the

opponent automaton plays after h_t.
If a leaf node is added, for in which this history h_t is not unique, i.e. for which $\lambda_{M_1}(i_1(h_{t-1})) \neq \lambda_{M_2}(i_2(h_{t-1}))$, this node is categorized as a revelation. Alternatively, if at this history h_t we revisit a combination of M_1's and M_2's state that we have visited before, then the node is categorized as a cycle. More precisely, this is the case if the considered node has an ancestral node that is d steps closer to the root and has a history h_{t-d} for which $i_1(h_t) = i_1(h_{t-d})$ and $i_2(h_t) = i_2(h_{t-d})$.
The algorithm terminates when there are no further uncategorised leaf nodes. The order in which leaf nodes are considered in the procedure until this point is reached, has no influence on the resulting tree.
Output: A tree in which all leaves are categorised as either a revelation or a cycle.

Clearly, Algorithm 2 terminates in finite time, as the maximum depth of the tree is bounded by the number of possible combinations of states of the two automata $n_{M_1} n_{M_2}$ that can be visited before one combination has to be repeated. Analogous to the arguments in Section 1, the motivation for stopping once a cycle has been reached is that there always exists a best-responding automaton that conditions its behaviour only on the opponent's (or here, the opponents') state, and not on the complete histories.

The key to our procedure for finding best responses in mixed populations is that we separate the play between the player we consider and their opponent into two phases: the phase before the opponent's type has been revealed, and the phase after. The possible dynamics in the phase before are fully described by the tree generated in Algorithm 2. Calculating payoffs along sequences on this tree is straightforward. Moreover, once the opponent's type is revealed, optimal play against this opponent can simply be identified with Algorithm 1. Based on this reasoning we can make the following observation.

Theorem 19. *In a mixed population of two finite state automata of sizes n_1 and n_2, there exists an automaton of size $n_1 + n_2 + (n_1 \cdot n_2)$ that plays optimally (plays a best response) in this mixed population.*

Proof. As argued in Section 1, the optimal stage game action at time t only depends on the opponent's state, and not on the complete history h_t. We can, thus, restrict attention to strategies that for a given state that their opponent is in (or a given combination of states that their potential opponents would be in) always play the same stage game action.
This leaves us with two cases. Case 1: It is optimal to play a sequence of actions that never reveals the type of the opponent. In that case this sequence can be represented with at most $(n_1 \cdot n_2)$ states. Case 2: It is optimal to play a sequence that does reveal the opponent's type. Here, the revelation must occur after at most $(n_1 \cdot n_2)$ rounds of play. After there revelation, we require at most an additional n_1 states to optimally respond to the first, and an additional n_2 states to optimally respond to the second automaton. □

The following algorithm finds such a best responding strategy.

Algorithm 3: Best Responder in Mixed Population

Input: $M_1 = \{\{1, ..., n_{M_1}\}, \lambda_{M_1}, \mu_{M_1}\}$, $M_2 = \{\{1, ..., n_{M_2}\}, \lambda_{M_2}, \mu_{M_2}\}$, p_1, δ, π

Procedure (Sketch): (1) Defining Blind Shot Action Set and Payoffs:

Generate a Blind Shot Tree (Algorithm 2). Define the action set A_0 as the collection of action-sequences along the simple paths from the root to the leaves of the Blind Shot Tree. For each $a_0 \in A_0$ (i.e. for each path from root to leaf), store the number $l(a_0)$ of edges in the path (i.e. the length of the history corresponding to that leaf-node). Next, calculate the expected payoff for each a_0[2]. We denote this payoff as $\pi_0(a_0)$. For leaf nodes categorized as cycles $\pi_0(a_0)$ takes on the discounted value of an infinite repetition of this history. For leaf nodes categorized as revelations, store furthermore the respective states i_1 and i_2 that automaton i_1 and i_2 find themselves in after the sequence. For ease of notation we simply refer to those as $\mu_M(a_0)$ for $a_0 \in A_0$.

(2) Best Responder Algorithm on Modified Action Set:

(2.1) Start with a $(1 + n_{M_1} + n_{M_2})$-element vector of zeros $V_0 = (0, \ldots, 0)$.

(2.2) In the i'th round of the algorithm:

- Solve

$$V_i(1) = \max_{a_0 \in A_0} \begin{cases} \pi_0(a_0) + \delta^{l(a_0)} \Big(p_1 V_{i-1}(1 + \mu_{M_1}(a_0)) \\ \qquad + (1 - p_1) V_{i-1}(1 + n_{M_1} + \mu_{M_2}(a_0)) \Big) & \text{if } a_0 \text{ results in a revelation} \\ \pi_0(a_0) & \text{if } a_0 \text{ results in a cycle} \end{cases}$$

- for every state k in $\{1, \ldots, n_{M_1}\}$, solve

$$V_i(1 + k) = \max_{a \in A} \big(\pi(a, \lambda_{M_1}(k)) + \delta V_{i-1}(1 + \mu_{M_1}(k, \lambda(k), a) \big)$$

- and for every state k in $\{1, \ldots, n_{M_2}\}$, solve

$$V_i(1 + n_{M_1} + k) = \max_{a \in A} \big(\pi(a, \lambda_{M_2}(k)) + \delta V_{i-1}(1 + n_{M_1} + \mu_{M_2}(k, \lambda(k), a) \big)$$

(2.3) Repeat step (2.2) until a round passes at which $V_i = V_{i-1}$.

Output: A set of optimal blind shot actions a_0^* and of optimal actions post revelation against either automaton in any of their respective states, and a vector of expected payoffs $V^* = V_i$. [3]

[2] For a_0 that are categorized as cycles, this is the discounted payoff of the infinitely repeated cycle, and the sequence of actions leading up to the cycle. For those a_0 that end in a revelation this is the sum of the discounted sums of payoffs for playing this sequence of actions against opponent of type 1 or of type 2 respectively, weighted by their respective frequencies p_1 and $1 - p_1$.

[3] The sequence of actions that are played pre-revelation a_0 can, of course, be represented as an automaton with $l(a_0) \leq n_{M_1} n_{M_2}$) states in accordance with Theorem 19.

Proof. **Correctness and Convergence of Algorithm 3**
Convergence of $V_i(k)$ for $k \geq 2$ is implied by the convergence of Algorithm 1. Now, as the calculations of all $V_i(k)$ for all i and all $k \geq 2$ are independent of the value of $V_{i-1}(1)$, there will be a round i at which all $V_i(k)$ with $k \geq 2$ have converged, and solving the maximization problem for $V_i(1)$ will yield a correct solution.

□

Generalising this approach to populations with more than two distinct strategies requires, firtstly, that we adapt the Blind Shot Tree Algorithm. Rather than distinguishing only between histories after which all automata would have played distinct sequences of actions, we must also distinctly label histories, after which different subsets of the different automata would have played distinct sequences of actions.

The resulting Blind Shot Tree, thus, specifies action sequences a_0 for which the time to revelation $l_j(a_0)$ depends on the type of the automaton j. Moreover, action sequences a_0 can end in different cycles for different subsets of the set of automata, and/or only reach revelations for some automata. Algorithm 3 would, thus, also have to be adapted slightly. Specifically, in step (2.2), for maximising $V_i(1)$, one needs to choose the a_0 that maximizes the sum the payoffs for playing a_0 against all automata in the population, weighted with their respective frequencies and the payoffs of future actions $\delta^{l_j(a_0)}(p_j V_{i-1}(\mu_{M_j}(a_0)))\big)$ for all automata j which are revealed when playing a_0.

Moreover for the j'th automaton and every state k in $\{1, \ldots, n_{M_j}\}$, step (2.2) would need to solve

$$V_i(1 + \sum_{m=1}^{j-1} n_{M_m} + n_{M_j} + k) = \max_{a \in A} \big(\pi(a, \lambda_{M_j}(k)) + \delta V_{i-1}(1 + \sum_{m=1}^{j-1} n_{M_m} + \mu_{M_j}(k, \lambda(k), a)\big).$$

Correctness and Convergence of Algorithm 3 are unaffected by this change.

4.4 Best responder algorithm in mixed populations with endogenous separation

In the previous section, by summarising all sequences of blind shots into one action set A_0, we have created a state-like description of the situation of being matched with a new random opponent in a mixed population, and proposed an algorithm (Algorithm 3) that assigns a value to this situation. We now extend this procedure to a setting with endogenous separation, by permitting

the transition functions of the respective automata to point to the pre-revelation state:[4]

$$\mu_{M_j} : \{1, \ldots, n_{M_j}\} \times A^2 \mapsto \{-\sum_{m=1}^{j-1} n_{M_m}, 1, \ldots, n_{M_j}\}$$

In order to apply Algorithm 3, this addition requires that we also permit sequences in the Blind Shot Tree to end in an endogenous separation. This requires only minor modifications to the implementation of Algorithm 2 (see code in Supplementary Materials for details).

Proof. **Correctness and Convergence of Algorithm 3 with Endogenous Separation**[5]
We apply the arguments that García and van Veelen (2018) use to prove correctness and convergence of Algorithm 1, with some very minor modifications. A solution for a_0^* is characterized by:

$$V^*(1) = \pi(a_0^*) + \left[\sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot V^*(\mu_{M_j}(a_0^*))\right]$$

To accommodate arbitrary numbers of automata j, we replace the notation $l(a_0)$ from the two-automata setting with $l(a_0|j)$. For each j, the expression $l(a_0|j)$ refers to the number of stage games until an automaton of type j can be uniquely identified under a_0, with $l(a_0|j) = \infty$ if j can never be uniquely identified under a_0.

Moreover since action a_0^* is available at every iteration of Algorithm 3, we have that:

$$V_i(1) \geq \pi(a_0^*) + \left[\sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot V_{i-1}(\mu_{M_j}(a_0^*))\right]$$

With these arguments we can approximate the difference between the value $V_i(1)$ at the i'th iteration and the value at the final iteration $V^*(1)$.

$$\begin{aligned} V^*(1) - V_i(1) &\leq \sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot V^*(\mu_{M_j}(a_0^*)) - \sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot V_{i-1}(\mu_{M_j}(a_0^*)) \\ &\leq \sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot \left(V^*(\mu_{M_j}(a_0^*)) - V_{i-1}(\mu_{M_j}(a_0^*))\right) \end{aligned}$$

Using that V_i increases weakly in every iteration of Algorithm 3, so that for every i, we have $V_i \leq V^*$, we can find majorants for the right hand side of this inequality, and show that the steps in Algorithm 3 constitute a contraction.

[4]Note that the index given to the pre-revelation state, ensures that the term $(1+\sum_{m=1}^{j-1} n_{M_m} + \mu_{M_j}(k, \lambda(k), a)$ in Algorithm 3 and its generalization takes on the value 1, i.e. the index of the pre-revelation state in the vector V.

[5]Convergence of $V_i(k)$ for $k \geq 2$ is now no longer independent of the convergence of $V_i(1)$, which means that the proof of correctness and convergence of Algorithm 3 from the previous section does not apply here.

$$
\begin{aligned}
V^*(1) - V_i(1) &\leq \sum_j p_j \cdot \delta^{l(a_0^*|j)} \cdot \left(V^*(\mu_{M_j}(a_0^*)) - V_{i-1}(\mu_{M_j}(a_0^*))\right) \\
&\leq \delta \left[\sum_j p_j \cdot \left(V^*(\mu_{M_j}(a_0^*)) - V_{i-1}(\mu_{M_j}(a_0^*))\right)\right] \\
&\leq \delta \max_k \left(V(k)^* - V_{i-1}(k)\right)
\end{aligned}
$$

For the remaining states $k \geq 2$ we can reason analogously that $V^*(k) - V_i(k) \leq \delta \left(V^*(k) - V_{i-1}(k)\right)$, this step is identical to the proof given in García and van Veelen (2018). Jointly these inequalities imply that

$$
\max_k \left(V^*(k) - V_i(k)\right) \leq \delta \max_k \left(V^*(k) - V_{i-1}(k)\right)
$$

Since $\delta < 1$ and since the difference $(V^* - V_i)$ is always non-negative, this implies that the the sequence of V_i converges to V^*.

□