Exploring the performance of a baseline for value based statistical algorithms on dynamic Iterated Prisoner's Dilemma networks

Muhan Li

Abstract

We conducted research on inspecting the performance of a value based statistical algorithm on a network where each agent pair plays an extension of the iterated prisoner's dilemma game (IPD), which supports choosing and refusing partners, also known as IPD/CR. Here we show by multi-agent simulation that this simple statistical algorithm has comparable performance to other well known strategies on IPD games such as tit-for-tat (TFT) and win-stay-lose-shift (Pavlov). We also add noise corruption to the network to simulate networks in real scenarios where agents may misunderstand the passed message, and we compare the robustness of several well-known IPD strategies as well as the proposed statistical algorithm

Keywords— Iterated prisoner's dilemma, evolution-of-cooperation, value-based algorithm, network

1 Introduction

Cooperative behaviors are commonly seen among individuals of species, more complex organizations may also exhibit cooperative behaviors such as the union of countries. For a long time, the iterated prisoner's dilemma game has became a test bed for the emergence of cooperative strategies due to its stricter punishment compared to non-iterated prisoner's dilemma games. In the non-iterated setting, defection is the dominate strategy even if the other player choose to cooperate, while the accumulative property of the iterated version has allows more cooperative, mutually benefitial strategies to occur.

In the iterated scenario, Axelrod et al. [1] mentioned supiority of the tit-for-tat (TFT) strategy, which is suprisingly simple enough to just copy the opponent's last action, over other intricate strategies, such as modeling the behavior of the opponent as a Markov process and then uses Bayesian inference to select the best choice in the long run. They also show that the good strategies possess some common charateristics such as preferring cooperation, relatiating the opponent when needed, forgiving the opponent after retaliation, etc. In [2] and [3] they provided details of tornament experiments and demonstrates the importance of assisting strategies refered to as "kingmakers" which helps cooperative strategies to perform better and punishes selfish strategies.

Forgel et al. [4] used genetic programming with finite state machines to search for the best performming strategy. Press et al. [5] approached the problem by modeling the play of two strategies as a Markov matrix with a stationary vector, and showed the linear dependence of their zero-determinant strategies. Some more recent studies [6], [7] have used traditional optimization methods such as partical swarm optimization, reinforcement-learning, etc. to find the dominate strategy.

It is a natural thought to extend iterated prisoner's dilemma game to more complex network structures, and analyze the performance of various algorithms, or the survival robustness of the whole network, many studies [8], [9], [10], [11] have already explored this area, performing experiments on various topology, mainly scale-free and small-world networks as they reflects the selection bias over target opponents, and the graph property constrains in real social networks.

Noise is also an import area of IPD strategy research since successful strategies must be robust when the input is corrupted with some kind of noise coming from natural randomness, such as guassian or uniform. The famous failure example mentioned in [12], [13] is the tit-for-tat strategy, since by letting TFT play with itself, a simple flipping-binary error would make one end think the other end has defected, and they will be stuck in the loop of "cooperation-defection, defection-cooperation". Kraines et al. [14] developed a stochastic family of strategies based on the Pavlov strategy, Wu et al. [15] proposed two relaxed versions of TFT using generosity and contrition. However, although generosity is a solution, but it is also a vulnerability under extreme payoff configurations since adversaries can use this time window of generosity to attack the generous strategy by always choose to defect.

In this paper, we will develop a simple stochastic strategy baseline named random table sampling (RTS), which could be considered as a relaxed version of the TFT strategy that also supports actively exploiting opponents by cheating. We will compare the performance of our model under different parameter settings with other famous deterministic models using pair-wise tournaments, with a sweep over network configurations and payoff configurations.

2 Backgrounds

2.1 PD, IPD, IPD/CR

The prisoner's dilemma (PD) game is a well known case being studied in the game theory, where two agents A and B may select an action from a binary action space $a_{defect}, a_{cooperate}$, and gets rewarded by their actions from a payoff matrix as shown in table (1).

Table 1: Payoff matrix of prisonner's dilemma game.

B A	cooperate	defect
cooperate	R R	T S
defect	S T	P P

For both players, if they choose to cooperate, each of them will receive R as reward, if one choose to cooperate and another choose to defect, the one which choose to cooperate will receive the temptation payoff denoted as T, and the cooperating one recieves "sucker's" payoff denoted as S, if both of them defect, they will receive the punishment payoff P.

The game is only meaningful if there exists a possibility to exploit the cooperation and receives a reward higher than choosing to cooperate on both sides, therefore condition (1) is required for the game to be a prisoner's dilemma instance:

$$T > R > P > S \tag{1}$$

Where $R > P$ indicates that mutual cooperation is superior to mutual defection and $T > R, P > S$ implies the exploitation case where defection dominates other policies for both agents.

The iterated prisoner's dilemma (IPD) game proposed by Axelrod et al. [1] extends the prisoner's dilemma game and allows more than one game to happen between two involving agents, agents need to hold a memory of previous opponent's actions and select an action according to the history, the iterative extension adds condition (2) to ensure mutual cooperation is more beneficial than alternating cooperation and defection.

$$2R > T + S \tag{2}$$

To capture the preferential selection of partners in networks involving more than 2 agents, Stanley et al. [16] proposed the choice and refusal mechanism (IPD/CR) characterized by five stages: choice, refusal, play, cleanup and update in each iteration, the choice stage send offers to other agents with high expected payoffs, the refusal stage rejects offers from agents with low expected payoffs, the play stage is equivalent to a prisoner's dilemma game, and the cleanup and update stage will update the internal payoff estimation of each agent. The exact execution detail could be represented as pseudo code (1). The meaning of each symbol are listed in table (2).

Table 2: Payoff matrix of prisonner's dilemma game.

Symbol	Meaning	Value
i	Iteration number.	$i = 1, 2, \cdots$
a_m	Agent with index m.	
$\pi^{i-1}(m\|n)$	Expected payoff to a_n from playing a PD game with a_m in iteration i.	
A	Set of all agents.	
E	Initial payoff estimation.	
T	Maximum iteration times.	
W	*Wallflower payoff* (default payoff when offer is rejected).	
K	Maximum number of opponents for each agent.	$1 \leq K \leq \|A\| - 1$
U	Reward entry from the payoff matrix.	
w	Memory weight of old payoff.	$0 \leq w < 1$
τ	Minimum acceptable payoff threshold.	
$\arg\max^K$	Input value of top K output values.	

Algorithm 1 IPD/CR

```
1: procedure IPD/CR($A, E, T, W, K, U, w, \tau$)
2:     for all $a_n \in A$ do                                   ▷ Initialize payoff estimations.
3:         for all $a_m \in A, m \neq n$ do
4:             $\pi^0(m|n) \leftarrow E$
5:         end for
6:     end for
7:     $i = 1$
8:     while $i < T$ do
9:         (For agent n)
10:        $M \leftarrow \arg\max_m^K \pi^{i-1}(m|n)$                       ▷ Choice stage.
11:        $O \leftarrow sendAndReceiveOffer(M)$
12:        $O' \leftarrow \{o|o \in O, \pi^{i-1}(o|n) > \tau\}$              ▷ Refusal stage.
13:        $R \leftarrow rewardOfPD(O')$                                ▷ Play stage.
14:        Fill array $R$ with $W$ for $O \setminus O'$                    ▷ Clean-up stage.
15:        $\pi^i(m|n) = \begin{cases} \pi^{i-1}(m|n), & \text{If rejected / rejected by m.} \\ w\pi^{i-1}(m|n) + (1-w)U, & \text{If played.} \end{cases}$    ▷ Update stage
16:    end while
17: end procedure
```

The original paper of IPD/CR cited the criterion filtering work of Tesfatsion et al. [17] and stated that the moving average estimation is a special case of criterion filtering which is a strongly consistent estimation for the true expected return function.

2.2 Non-trivial networks

Apart from common network topologies like grid/torus, star, wheel, ring or fully-connected, random networks deserve more attention since they may approximate real social networks, the main testing ground for IPD, much better than previously mentioned fixed topologies. In this section, we will talk about the definition and special attributes of random networks used in this study.

2.2.1 Erdős - Rényi model

There are two definitions of the Erdős - Rényi model, the first one is the $G(n, M)$ model, where a graph is uniformly chosen from the random collection of all graphs with n nodes and M edges, the second model is the $G(n, p)$ model, where for each pair of nodes the probability of existing an edge between them is p. Therefore, the probability of generating a graph $G(|V| = n, |E| = M)$ is binomial:

$$P\{G(|V| = n, |E| = M)\} = p^M (1-p)^{\binom{n}{2}-M} \tag{3}$$

In the 1960 paper of Erdős and Rényi [18] they examined the properties of the $G(n, p)$ model, one of them is: if $p > \frac{c \ln n}{n}, c > 1$, then a graph randomly sampled from the $G(n, p)$ model will be connected (i.e. only have one connected component). This attribute is important in our research as we wish there is no isolated components in the generated graph, if not satisfied, the simulation may degenerate to computing several smaller simulations with less agents.

2.2.2 Watts–Strogatz model

The Watts–Strogatz model [19] was designed to address the lack of clustering problem in the ER model, since the probability of connecting any pair of two nodes are independent, local nodes are not more likely to be joined in an existing connected component when we generate the graph in a sequential way. Let K be the size of neighbors on the both sides of node from a ring network, the generation procedure first connects all $|V|$ nodes to its K neighbors, and then rewire the connection of all nodes with probability β while avoiding self loops and duplication.

There are two main properties used to characterize the whole spectrum of network families generated with $\beta \in [0, 1]$, the *characteristic path length* L and the *clustering coefficient* C. Let graph $G = (V, E)$, and $g(i, j) = \mathbb{I}\{\exists e \ inE, e = (i, j)\}$ be the adjacency function for node i and j, $d(i, j)$ be the shortest path function, then the characteristic path length is defined as equation (4).

$$L = \frac{1}{|V|(|V|-1)} \sum_i \sum_{j \neq i} d(i, j) \tag{4}$$

Let $N^1(i) = \{k \in V | g_{i,k} = 1\}$ be the set of nodes directly adjacent to i, and $E(i) = \{(j, k) g(j, k) = 1, k \neq j, j \in N^1(i), k \in N^1(i)\}$ be the set of edges connecting neighboring nodes of node i, the clustering coefficient is defined as equation (5).

$$C = \frac{1}{|V|} \sum_{i} \frac{|E(i)|}{|N(i)|(|N(i)| - 1)/2} \tag{5}$$

The spectrum of all generated graphs could be visualized by figure 1. For $\beta = 0$, there is no rewiring and a regular network with $deg(v) = K, \forall v \in V$ is generated, both L and C are large, for $\beta = 1$, a random network is generated and L and C are small. The networks generated when $0 < beta < 1$ are usually referred to as small world networks, where L is small and C is large. The small world model can approximate the clustering and short-path properties of real social networks, but does not conform to the power-law required by the scale-free property.

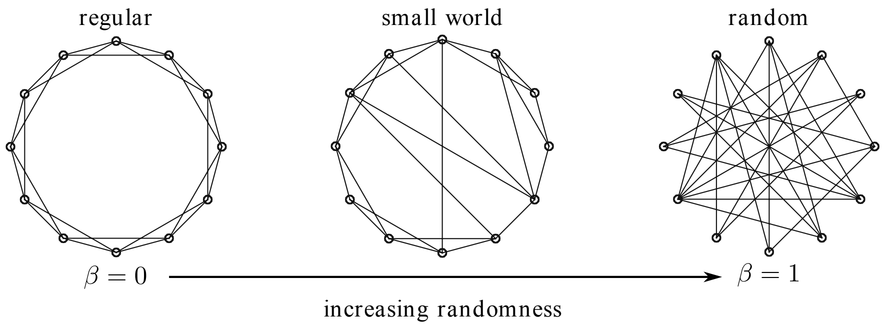

Figure 1: Network types from $\beta = 0$ to $\beta = 1$, Adapted from figure 1 of [10]

2.2.3 Barabási-Albert model

Barabási et al. proposed a new model in [20] which satisfies the power law on node degree distribution described by equation (6).

$$P\{v \in V, deg(v) = k\} = ck^{-\gamma} \tag{6}$$

Where c and γ are constants. This model is also commonly referred to as the preferential attachement model due to the nature of its generation algorithm. A fitness function is defined for all nodes, usually the degree is used as the fitness function, we start with m_0 fully-connected nodes and iteratively add new nodes and connections by equation (7) and (8), let V' be the set of already existing nodes, v be the newly added node, V^* be the set of chosen nodes, $m, m <= m_0$ the number attached nodes, and $\arg\max^m$ for top m values.

$$P\{(v, v'), v' \in V\} = \frac{deg(v')}{\sum_{v' \in V} deg(v')} \tag{7}$$

$$V^* = arg\, max_{v'}^{m} P\{(v, v'), v' \in V\} \tag{8}$$

3 Model

3.1 Random table sampling

3.1.1 Naive version

In reinforcement learning literature, many famous algorithms are designed to maintain a value estimation function over the action space, such as Q-learning [21], and other famous derivatives [22] [22] [23] which incorporates the idea of Q-learning to some extent.

The same idea could be used to create a simple value based algorithm, For every connected pair of agent m and n we first define an estimator for the cooperation and defection probability using the moving average method as recommended by [17]. We left out the superscript mn for simplicity. The definition is equation (9)

$$\begin{aligned}
\tilde{P}_0 &= \begin{bmatrix} \tilde{P}_0\{act = cooperate\} \\ \tilde{P}_0\{act = defect\}) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
\tilde{P}_t &= \begin{cases} w\tilde{P}_{t-1} + (1-w)\begin{bmatrix} 1 \\ 0 \end{bmatrix} & , \text{ If } act_t = cooperate \\ w\tilde{P}_{t-1} + (1-w)\begin{bmatrix} 0 \\ 1 \end{bmatrix} & , \text{ If } act_t = defect \end{cases}
\end{aligned} \tag{9}$$

Where t is the iteration number from $1, \cdots$ .... We initialize the inital opponent cooperation probability estimation as 1 and defection probability as 0 to make the model prefer cooperation initially. Then at iteration t, we can then use the payoff matrix to compute expected rewards under 2 actions as equation (10)

$$\begin{aligned}
\mathbb{E}_{cooperate} \simeq \tilde{\mathbb{E}}_{cooperate} &= \begin{bmatrix} R & S \end{bmatrix} \tilde{P}_{t-1} \\
\mathbb{E}_{defect} \simeq \tilde{\mathbb{E}}_{defect} &= \begin{bmatrix} T & P \end{bmatrix} \tilde{P}_{t-1}
\end{aligned} \tag{10}$$

Then we may immediately propose the naive algorithm (2) using two expectation values. We name the algorithm as ”random table sampling” since action is randomly selected from a vector of distribution.

We use the constant parameter $\theta \geq 0$ to control ”hardness” of the softmax function, when $\theta \to 0$, the softmax output approaches equal probability for both actions, and when $\theta \to \infty$, $softmax = \max$. Vector Φ is used to control model's preference over rewards, when set to $\begin{bmatrix} 1 & 1 \end{bmatrix}$, the model is unbiased when choosing its action using the expected rewards.

However there are several problems with this naive version:

- **Preference is hard to control**: Since the payoff matrix is unnormalized, under extreme cases like $T \gg R$,

Algorithm 2 RTS-naive

```
1: procedure RTS-NAIVE($\theta, \Phi$)
2:     $\tilde{P}_0 \leftarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
3:     while $i < T$ do
4:         (For agent $n$ and $m$)
5:         $\tilde{\mathbb{E}} \leftarrow \begin{bmatrix} R & S \\ T & P \end{bmatrix} \tilde{P}_{i-1}$
6:         $P_{act} \leftarrow softmax(\theta \Phi^T \tilde{\mathbb{E}})$    ▷ Compute action distribution
7:         $act \sim P_{act}$    ▷ Sample action.
8:         $act_o = playPD(act)$    ▷ Plays PD game with opponent and gets opponent's action.
9:         $\tilde{P}_i = w\tilde{P}_{i-1} + (1-w)act_o$    ▷ Update probability estimation
10:    end while
11: end procedure
```

the preference vector Φ is not compatible with other cases. Preference over cooperation is also conflicting with necessary defection (i.e. defect when the opponent is highly likely to defect).

- **Short-sighted and selfish**: The algorithm is reward based but doesn't take long term rewards into account, even if we know that continuous cooperation is better than alternating cooperation-defection. It is also selfish since it chooses action with the maximum return, under PD settings where $T > R$, it will never cooperate without the preference modifier Φ.

3.1.2 Advantage of defection

A successful algorithm may choose to defect and exploit opponent's cooperation sometimes, but not frequently, otherwise the opponent will retaliate with defection. The algorithm must choose to defect and defend itself when opponent is an enemy with high probability of defection, therefore we can separate defection into 2 cases:

- **Defection for exploitation**: Choose to defect to gain extra benefit $T - R$ when opponent is cooperative.
- **Defection for self-defense**: Choose to defect to prevent losing extra benefit $P - S$ when opponent is invasive.

The first case is optional while the second case is necessary. Therefore we can represent the expected advantage of defection over cooperation as equation 11

$$\mathbb{E}_{adv} = \mathbb{E}_{defect} - \mathbb{E}_{cooperate} = p(T-R) + (1-p)(P-S) \tag{11}$$

Where p is the true cooperation probability of the opponent, which could be approximated using previously defined estimator $\tilde{P}$. We wish to define an advantage function whose output is invariant of payoff configuration, therefore, we could normalize the expected advantage by the largest advantage difference in the payoff matrix:

$$f_{adv} = \frac{\mathbb{E}_{adv}}{\max\{T-R, P-S\}} \tag{12}$$

We can then analyze the output of this advantage function under 3 payoff configurations:

- $T - R \gg P - S$, the *speculator* case, since exploitation yields much higher returns that compensates the loss of opponent's defections in the future.
- $T - R \ll P - S$, the *defender* case, since being exploited is a much worse situation, the algorithm must defend itself by defection.
- $T - R \simeq P - S$, the *Candide* case, also the most common case, since exploitation doesn't compensates the extra loss of being defended (choose to cooperate, but opponent never trusts and always choose to defect). The algorithm should stick to cooperation and rarely exploits.

The behavior of $f_a dv$ is linear with regard to p in these 3 cases, in figure (2) we visualize f_{adv} as a function of opponent cooperating probability p.

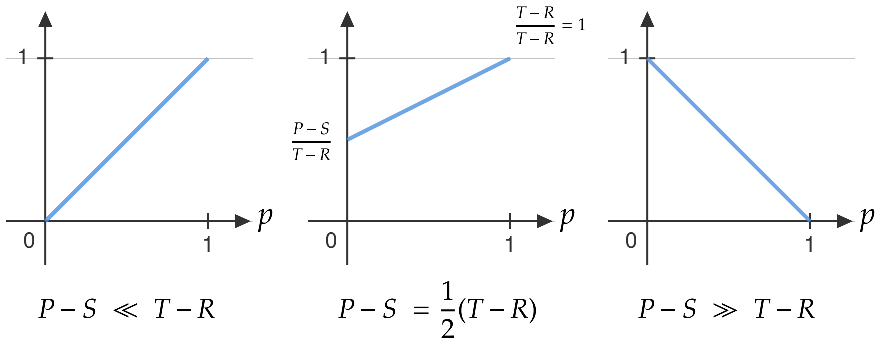

Figure 2: Visualization of f_{adv} in 3 payoff cases.

In the special case where $T - R = P - S$, $f_a dv$ becomes $y = 1, 0 \leq x \leq 1$, this is interpretable since when $p < 0.5$ (opponent is likely to defect), we choose to also defect with probability equal to $f_a dv = 1$, when $p > 0.5$ (opponent is likely to cooperate), we choose to defect since defection is the dominant strategy in PD, in IPD we need to take long-term rewards into consideration, which we will address in the second version of the RTS algorithm.

3.1.3 RTS with advantge of defection

Now with the advantage function of defection defined, we can propose a second version of RTS, described in algorithm (3), we also include preference parameter Φ, but is a scalar in the second version, to adjust advantage of exploitation. The new advange function f^{Φ}_{adv} is defined as equation (13).

$$f^{\Phi}_{adv} = \frac{p\Phi(T-R) + (1-p)(P-S)}{\max\{\Phi(T-R), P-S\}} \tag{13}$$

The new algorithm with the exploitation scaling factor Φ can be "cunning", when Φ is appropriate and small enough, it will sometimes cheat the opponent, but keep cooperating mostly, in the most common *Candide* case mentioned above, the probability of defecting for exploitation when $p \to 1$ is approximately equal to Φ. In most cases, it will only defect for self-defense, when $p \to 0$.

When $\Phi = 0$ and memory weight of old probability estimation $w = 0$, RTS-advantage is equivalent to the tit-for-tat strategy, since it will always cooperate, and only defect when provoked by the opponent. When w is appropriate,

Algorithm 3 RTS-advantage

```
 1: procedure RTS-ADVANTAGE($\Phi$)
 2:     $\tilde{P}_0 \leftarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
 3:     while $i < T$ do
 4:         (For agent $n$ and $m$)
 5:         $P_{defect} \leftarrow f_a^{\Phi} dv(\tilde{P}_{i-1})$        ▷ Compute probability of defection
 6:         $act \sim \{P_{defect}, 1 - P_{defect}\}$        ▷ Sample action using a two-point distribution.
 7:         $act_o = playPD(act)$        ▷ Plays PD game with opponent and gets opponent's action.
 8:         $\tilde{P}_i = w\tilde{P}_{i-1} + (1-w)act_o$        ▷ Update probability estimation
 9:     end while
10: end procedure
```

RTS-advantage is similar to relaxed families of the tit-for-tat strategy, such as tit-for-2-tats, which only defect when last two actions of the opponent is defection.

3.2 Network simulation

Multi-agent simulation is a popular way of conducting experiments to gather empirical data for algorithm performance analysis and comparison. Almost all researches [1] [2] [12] [4], on IPD are experimented with multi-agent modeling. Nowadays we have powerful multi-agent modeling tools such as NetLogo [24], we abstract agents with different policies as turtle agents of breed "act-agents" in NetLogo, and their interaction edge as link agents. An example of the generated graph is presented in figure (3)

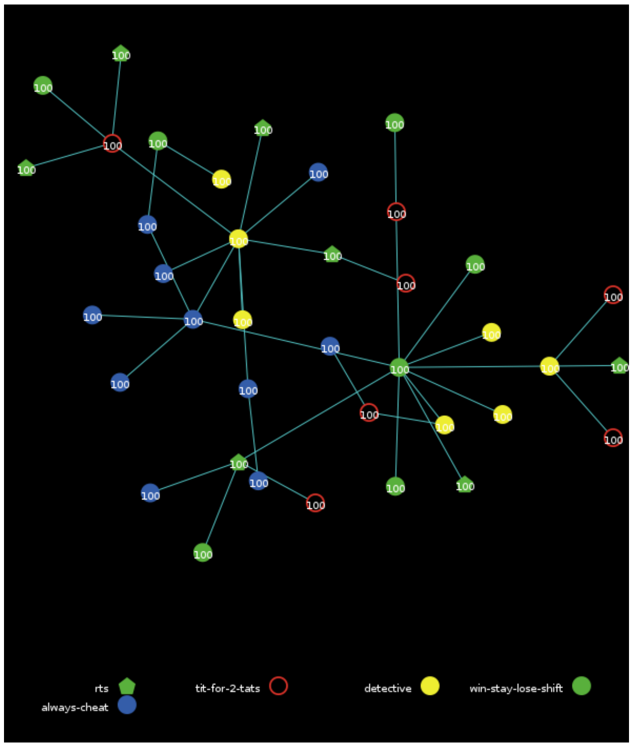

Figure 3: An example of the generated graph, Barabási-Albert model[20] is used to generate the network.

3.2.1 Simplification of choice and refusal

Here we provide a simplified version of the choice and refusal mechanism in IPD/CR [16], the original mechanism involves 2 stages, choice and refusal to select opponents in each iteration, which is effectively equivalent to acting on a dynamically generated Barabási-Albert model[20] in each iteration, since we would like to experiment on different network models, we force agents to play with all neighboring agents on the graph, and define the *trust* mechanism to sever / generate new links.

Here we define trust of node v_i on other adjacent nodes v_j as a moving average of the indicator function of cooperative actions, the exact definition is as equation (14), we use the same symbols, iteration number as t, .

$$f_{trust}^{t}(v_j) = wf_{trust}^{t-1}(v_j) + (1-w)\mathbb{I}\{act_j^{t-1} = cooperate\} \tag{14}$$

Then in each iteration we server the link if $f_{trust}^{t}(v_j) < \epsilon$, or the strategy decides to cut.

We defined several ways to generate new links when old ones are severed:

- **Disabled**: never generate new links.
- **Enable-disable**: When trust of links fall below threshold ϵ, the trust value is set to 0 for both sides and links are disabled, then in each iteration, they are restored by a value (0.05 in this case) until they reach ϵ, then links are restored.
- **By distance**: When links are severed, ask both sides to find their geographically furthest / nearest agent and connect to them.
- **By degrees**: When links are severed, ask both sides to find globally most-connected / least-connected agent and connect to them.
- **Random**: When links are severed, ask one side / both sides to find a random partner.

All methods avoids creating duplicate links and restore links severed in this turn. The random method is most interesting, it creates a survival bias of agents who maintain higher trust levels.

The explanation is straightforward. First, with more links, agents are more likely to survive, in the one side case, agents who maintain higher opponent trust will preserve their link with high-probability, while selfish agents with lower opponent trust will loss their links, while in the two side case, agents with higher trust will gets more and more links while agents with lower trust has near constant number of links.

3.2.2 Transmission noise

Bendor et al. [25] used a continuous metric to reflect the cooperation level, and apply a continuous uniform random variable to perturb the decision process, since some of the evaluated strategies are discrete in nature, we instead opt to apply transmission noise to the eventual result, i.e. first determine flipping or not by probability p_noise, then randomly select one side, and flip its action from cooperate to defect, or defect to cooperate.

3.3 Simulation

Compared to tournaments in [2] [3], Our simualtion model is different in 3 aspects:

- **No round-robin**: All PD games represented by each edge are run in parallel.
- **No restriction on edge life**: Edges persist as long as nodes survive and not severed by the trust mechanism.
- **Nodes are limited by fitness**: Each node are initialized with a fitness value of 100, in each iteration the fitness value is reduced by a constant, if it is below 0, they will die and optionally be replaced by other strategies if allowed.

4 Experiments

4.1 Test setup

We use the same benchmark criteria as in [1] to test the performance of RTS in three dimensions, since exact definitions for these criteria are not exactly defined in the original paper, we formalize them as follows:

- **Robustness**: The count ot energy sum of a policy when tested with other policies after a fixed number of iterations.
- **Stability**: The ability of a policy to maintain or increase its ratio when invaded by another policy with a smaller initial ratio.
- **Initial viability**: The ability of a policy to increase its ratio when competing with the all-defection policy initialized with a larger ratio.

Since we also would like to benchmark the noise resistance ability of the RTS algorithm, we add another criterion:

- **Noise resistance**: The ability of an algorithm to prevent a homogeneous network of its own type from collapsing under the defender game in table 3b.

We test all algorithms on the three types of game payoff matrices defined previously, the value of each action is shown in table (3), policy and their respective ratio are shown in table (4), other configurations are shown in table (5).

For stability testing, we choose the always-cheat policy as adversaries since initially cooperative deterministic policies will always cooperate and has the same result. For initial viability we also use always-cheat to test whether specified policies can survive in a initially non-cooperative environment.

Since the model supports a wide range of parameters, it is not feasible to test model performance on every possible combination, we prune the parameter space by selecting ones that approximates real situations the best and remove others.

Table 3: Payoff matrix of tested environments

(a) Candide

A \ B	cooperate	defect
cooperate	3 \ 3	0 \ 5
defect	5 \ 0	1 \ 1

(b) Defender

A \ B	cooperate	defect
cooperate	3 \ 3	-20 \ 15
defect	-20 \ 15	1 \ 1

(c) Speculator

A \ B	cooperate	defect
cooperate	10 \ 10	-1 \ 20
defect	-1 \ 20	0 \ 0

Table 4: Policy composition of tested tasks.

Task	Composition	Metrics	Test times
Robustness	rts, tit-for-tat, tit-for-2-tats, win-stay-lose-shift, always-cheat, equal ratio	Policy count, Policy mean energy	5
Stability	tit-for-tat: 0.9, always-cheat: 0.1	Policy count	5
Stability	tit-for-2-tats: 0.9, always-cheat: 0.1	Policy count	5
Stability	win-stay-lose-shift: 0.9, always-cheat: 0.1	Policy count	5
Stability	rts: 0.9, always-cheat: 0.1	Policy count	5
Initial viability	tit-for-tat: 0.1, always-cheat: 0.9	Policy count	5
Initial viability	tit-for-2-tats: 0.1, always-cheat: 0.9	Policy count	5
Initial viability	win-stay-lose-shift: 0.1, always-cheat: 0.9	Policy count	5
Initial viability	rts: 0.1, always-cheat: 0.9	Policy count	5
Noise resistance	tit-for-tat: 1	Policy count	5
Noise resistance	tit-for-2-tats: 1	Policy count	5
Noise resistance	win-stay-lose-shift: 1	Policy count	5
Noise resistance	rts: 1	Policy count	5

4.2 Robustness

We visualize the number of times each algorithm has been ranked, either by their eventual number at test end, or their mean energy per agent at test end.

The rank by count is shown in table (6), and the rank by energy is shown in table (7), a smaller rank position is better than a larger rank position. RTS is worse than the three deterministic baselines: tit-for-tat, tit-for-2-tats, and win-stay-lose-shift in terms of number of eventual count, but has approximately the same or better performance as tit-for-tat in terms of energy. An interesting phenomenon is that the always-cheat strategy performs the best in the defender game setting, since defending could be quite hard due to extreme payoff settings, the always-cheat strategy could attack other strategies easily.

Table 5: Other configurations.

Config	All tested values
Agent number	100
Energy consumption	1.0
Dead mutation	allowed and mutate by count
Initial network type	[random, small-world, preferential-attachement]
New links formation	most-connected
Minimum link maintenance trust	[0, 0.5]
Transmission noise	[0, 0.1]
RTS old memory weight w	[0, 0.95]
RTS exploitation Φ	0

Table 6: Times of rank (by count) taken by each algorithm

(a) Candide

	rank-1	rank-2	rank-3
always-cheat	1	3	1
rts	1	2	18
tit-for-tat	21	1	2
tit-for-2-tats	1	0	2
win-stay-lose-shift	0	18	1

(b) Defender

	rank-1	rank-2	rank-3
always-cheat	19	3	2
rts	1	4	4
tit-for-tat	4	12	6
tit-for-2-tats	0	5	11
win-stay-lose-shift	0	0	0

(c) Speculator

	rank-1	rank-2	rank-3
always-cheat	1	1	2
rts	0	2	15
tit-for-tat	17	3	3
tit-for-2-tats	3	1	2
win-stay-lose-shift	3	17	2

Table 7: Times of rank (by energy) taken by each algorithm

(a) Candide

	rank-1	rank-2	rank-3
always-cheat	0	0	0
rts	2	6	8
tit-for-tat	4	8	9
tit-for-2-tats	14	6	3
win-stay-lose-shift	4	4	4

(b) Defender

	rank-1	rank-2	rank-3
always-cheat	9	1	3
rts	6	3	3
tit-for-tat	6	5	8
tit-for-2-tats	3	12	4
win-stay-lose-shift	0	3	5

(c) Speculator

	rank-1	rank-2	rank-3
always-cheat	0	0	0
rts	8	5	8
tit-for-tat	5	3	9
tit-for-2-tats	7	10	4
win-stay-lose-shift	4	6	3

4.3 Stability

In stability benchmark each strategy is tested against a small portion of always-cheat strategy, results of this experiment is shown in table (8), ”success ratio” is defined as the ratio of trials where the tested strategy has a higher or the same ratio when compared to its initial ratio. The other metric ”average number change” is the ratio between strategy’s eventual count divided by its initial number. RTS has similar performance as tit-for-tat and tit-for-2-tats in the speculator and candide game setting, but has mixed results in the defender game setting. While the success ratio of RTS is higher than win-stay-lose-shift and tit-for-2-tats, its average number change is the lowest, therefore in some settings it has failed and replaced by the always-cheat strategy.

Although win-stay-lose-shift performs the worst with regard to success ratio, and failed all competitions with always-cheat in the defender setting, we still would like to highlight its result on the average number change metric as it reflects its general stability on all cases, showing that it still has reasonable performance.

Table 8: Stability result of each algorithm.

(a) Candide

policy name	success ratio	average number change
tit-for-tat	1.0	1.0
tit-for-2-tats	1.0	1.0
win-stay-lose-shift	0.833	0.997
rts	0.958	0.999

(b) Defender

policy name	success ratio	average number change
tit-for-tat	0.5	0.5
tit-for-2-tats	0.125	0.98
win-stay-lose-shift	0.0	0.623
rts	0.25	0.314

(c) Speculator

policy name	success ratio	average number change
tit-for-tat	0.917	1.027
tit-for-2-tats	0.917	1.024
win-stay-lose-shift	0.833	0.998
rts	0.917	1.012

4.4 Initial viability

For initial viability experiments, we reuse the metrics from stability testing as the only difference between these two experiments is the initial ratio. The result is shown in table (9). RTS has medium performance approximately the same as tit-for-tat and tit-for-2-tats, and better than win-stay-lose-shift in all configurations. All algorithms fail horribly in the Defender setting since the ”sucker’s loss” is high enough to make all initially cooperative algorithms fail in a few iterations.

Table 9: Initial viability result of each algorithm.

(a) Candide

policy name	success ratio	average number change
tit-for-tat	0.667	0.895
tit-for-2-tats	0.5	0.769
win-stay-lose-shift	0.333	0.641
rts	0.5	0.791

(b) Defender

policy name	success ratio	average number change
tit-for-tat	0.0	0.177
tit-for-2-tats	0.0	0.068
win-stay-lose-shift	0.0	0.0
rts	0.0	0.079

(c) Speculator

policy name	success ratio	average number change
tit-for-tat	0.5	1.838
tit-for-2-tats	0.542	1.405
win-stay-lose-shift	0.292	0.822
rts	0.333	1.279

Table 10: Noise resistance result of each algorithm.

(a) Candide

policy name	average number change
tit-for-tat	1.0
tit-for-2-tats	1.0
win-stay-lose-shift	1.0
rts	1.0

(b) Defender

policy name	average number change
tit-for-tat	1.0
tit-for-2-tats	1.0
win-stay-lose-shift	1.0
rts	1.0

(c) Speculator

policy name	average number change
tit-for-tat	1.0
tit-for-2-tats	1.0
win-stay-lose-shift	1.0
rts	1.0

4.5 Noise resistance

In the noise resistance test we benchmarked all strategies by comparing their even number to the initial number, result is shown in table (10). An algorithm performing reasonably well should have ”average number change” be close to or above 1.0, surprisingly all algorithms perform very well in this test, however, it is observed that tit-for-tat will collapse after enough iteration in inidividual testing, therefore, we suspect the iteration times used by the test is insufficient to reveal the performance of algorithms.

5 Conclusions

In this paper we proposed a expected value based algorithm named random table sampling as baseline for other more complex stochastic algorithms also using some form of value metric. The model demonstrates comparable robustness, stability and initial viability performance as the famous tit-for-tat (TFT) strategy. Its noise resistance is better that tat-for-tat but still succumb to noise due to the accumulation effect of destabilization, the trust mechanism is shown to be able to stablize the network regardless of the strategy employeed in the noise resistance test. The win-stay-lose-shift and tit-for-2-tats, a relaxed version of TFT, has the best noise resistance performance, but could be exploited by RTS when the cheat preference ratio Φ is proper.

The same model could be reused to study problems more related to the macro-level the network structure:

- Will the ”kingmaker” strategy mentioned in [2] and [3] help policies like RTS and TFT to work better on networks instead of round-robin tournaments?
- Will the ”kingmaker” strategy help to stablize the network by spreading trust?
- Currently no algorithms can support automatic link severing without the trust mechanism, and the trust mechanism is slow, non-adaptive to reward S in the payoff matrix, can we implement a more efficient, adaptive trust mechanism, or add such functions to the strategy, and prevent the whole network from collapsing from distrust?

Such studies could be useful for securing the network from adversaries who are trying to disintegrate by spreading misinformation, or aiming at exploitation. Previous studies mainly focus on the performance of individual strategies rather than considering the ensemble of all policies as a whole on a complex network structure, therefore there is sufficient reason to claim that future researches can be conducted on a macro level rather than the micro level.

References

[1] R Axelrod and WD Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.

[2] Robert Axelrod. Effective choice in the prisoner's dilemma. *Journal of conflict resolution*, 24(1):3–25, 1980.

[3] Robert Axelrod. More effective choice in the prisoner's dilemma. *Journal of conflict resolution*, 24(3):379–403, 1980.

[4] David B Fogel. Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, 1(1):77–97, 1993.

[5] William H Press and Freeman J Dyson. Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413, 2012.

[6] Nelis Franken and Andries Petrus Engelbrecht. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on evolutionary computation*, 9(6):562–579, 2005.

[7] Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E Glynatsi, and Owen Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *PloS one*, 12(12):e0188046, 2017.

[8] Mark D Smucker, E Stanley, and Dan Ashlock. Analyzing social network structures in the iterated prisoner's dilemma with choice and refusal. *arXiv preprint adap-org/9501002*, 1995.

[9] Ya-Shan Chen, Hai Lin, and Chen-Xu Wu. Evolution of prisoner's dilemma strategies on scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 385(1):379–384, 2007.

[10] Masahiro Ono and Mitsuru Ishizuka. Prisoner's dilemma game on network. In Dickson Lukose and Zhongzhi Shi, editors, *Multi-Agent Systems for Society*, pages 33–44, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[11] Dandan Li, Jing Ma, Dun Han, Mei Sun, Lixin Tian, and H Eugene Stanley. The co-evolution of networks and prisoner's dilemma game by considering sensitivity and visibility. *Scientific reports*, 7(1):1–9, 2017.

[12] Robert Axelrod and Douglas Dion. The further evolution of cooperation. *Science*, 242(4884):1385–1390, 1988.

[13] Robert M May. More evolution of cooperation. *Nature*, 327(6117):15–17, 1987.

[14] David Kraines and Vivian Kraines. Learning to cooperate with pavlov an adaptive strategy for the iterated prisoner's dilemma with noise. *Theory and Decision*, 35(2):107–150, 1993.

[15] Jianzhong Wu and Robert Axelrod. How to cope with noise in the iterated prisoner's dilemma. *Journal of Conflict resolution*, 39(1):183–189, 1995.

[16] E Stanley, Dan Ashlock, and Leigh Tesfatsion. Iterated prisoner's dilemma with choice and refusal of partners. 1993.

[17] Leigh Tesfatsion. Direct updating of intertemporal criterion functions for a class of adaptive control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(3):143–151, 1979.

[18] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[19] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[20] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[21] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[22] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[23] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[24] U. Wilensky. Netlogo. http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.

[25] Jonathan Bendor, Roderick M Kramer, and Suzanne Stout. When in doubt... cooperation in a noisy prisoner's dilemma. *Journal of conflict resolution*, 35(4):691–719, 1991.