Five rules for friendly rivalry in direct reciprocity

Yohsuke Murase
RIKEN Center for Computational Science, Kobe, Hyogo 650-0047, Japan

Seung Ki Baek
Department of Physics, Pukyong National University, Busan 48513, Korea

Abstract
Direct reciprocity is one of the key mechanisms accounting for cooperation in our social life. According to recent understanding, most of classical strategies for direct reciprocity fall into one of two classes, ‘partners’ or ‘rivals’. A ‘partner’ is a generous strategy achieving mutual cooperation, and a ‘rival’ never lets the co-player become better off. They have different working conditions: For example, partners show good performance in a large population, whereas rivals do in head-to-head matches. By means of exhaustive enumeration, we demonstrate the existence of strategies that act as both partners and rivals. Among them, we focus on a human-interpretable strategy, named ‘CAPRI’, which is described by five simple rules. Our evolutionary simulation shows excellent performance of CAPRI in a broad range of environmental conditions.

Keywords: Social dilemma, Direct reciprocity, Evolutionary game

1. Introduction

Theory of repeated games is one of the most fundamental mathematical framework that has been long been studied for understanding how and why cooperation emerge in human and biological communities. Even when cooperation cannot be a solution of a one-shot game, repetition can enforce cooperation between the players by taking into account the possibility of
future encounters. A spectacular example is the prisoner’s dilemma (PD) game: It describes a social dilemma between two players, say, Alice and Bob, in which each player has two options ‘cooperation’ (c) and ‘defection’ (d). The payoff matrix for the PD game is defined as follows:

\[
\begin{pmatrix}
  \text{Bob} \\
  \text{Alice}
\end{pmatrix}
\begin{pmatrix}
  c & d \\
  (R, R) & (S, T) \\
  (T, S) & (P, P)
\end{pmatrix},
\]

where each entry shows (Alice’s payoff, Bob’s payoff) with \( T > R > P > S \) and \( 2R > T + S \). If the game is played once, mutual defection is the only equilibrium because Alice maximizes her payoff by defecting no matter what Bob does. However, if the game is repeated with sufficiently high probability, cooperation becomes a feasible solution because the players have a strategic option that they can reward cooperators by cooperating and/or they can punish defectors by defecting in subsequent rounds. This is known as direct reciprocity, one of the most well-known mechanisms for the evolution of cooperation (Nowak, 2006).

Through a series of studies, recent understanding of direct reciprocity proposes that most of well-known strategies act either as partners or as rivals (Hilbe et al., 2015, 2018a). Partner strategies are also called ‘good strategies’ (Akin, 2015, 2016), and rival strategies have been described as ‘unbeatable’ (Duersch et al., 2012), ‘competitive’ (Hilbe et al., 2015), or ‘defensible’ (Yi et al., 2017; Murase and Baek, 2018). Derived from our everyday language, the ‘partner’ and ‘rival’ are defined as follows. As a partner, Alice aims at sharing the mutual cooperation payoff \( R \) with her co-player Bob. However, when Bob defects from cooperation, Alice will punish Bob so that his payoff becomes less than \( R \). In other words, for Alice’s strategy to be a partner, we need the following two conditions: First, \( \pi_A = \pi_B = R \) when Bob applies the same strategy as Alice’s, where \( \pi_A \) and \( \pi_B \) represent the long-term average payoffs of Alice and Bob, respectively. Second, when \( \pi_A \) is less than \( R \) because of the Bob’s defection from mutual cooperation, \( \pi_B \) must also be smaller than \( R \), whatever Bob takes as his strategy. It means that one of the best responses against a partner strategy is choosing the same partner strategy so that they form a Nash equilibrium. If a player uses a rival strategy, on the other hand, the player aims at a payoff higher than or equal to the co-player’s regardless of the co-player’s strategy. Thus, as long as Alice is a rival, it is guaranteed that \( \pi_A \geq \pi_B \). Note that these
two definitions impose no restriction on Bob’s strategy, which means that the inequalities are unaffected even if Bob remembers arbitrarily many previous rounds.

Which of these two traits is favoured by selection depends on environmental conditions, such as the population size $N$ and the elementary payoffs $R$, $T$, $S$, and $P$. For instance, a large population tends to adopt partner strategies when $R$ is high enough. A natural question would be on the possibility that a single strategy is both a partner and a rival simultaneously. Let us call such a strategy a ‘friendly rival’ hereafter. Tit-for-tat (TFT) or Trigger strategies can be friendly rivals in an ideal condition that the players are free from implementation error due to “trembling hands”. However, this is not the case in a more realistic situation in which actions can be misimplemented with probability $e > 0$. Here, the apparent contradiction between the notions of a partner and a rival is seen as the most acute form. That is, Alice must forgive Bob’s erroneous defection to be a partner and punish his malicious defection to be a rival, without knowing Bob’s intention. This is the crux of the matter in relationships.

In this work, by means of massive supercomputing, we show that a tiny fraction of friendly rival strategies exist among deterministic memory-three strategies for the iterated PD game without future discounting. Differently from the strategies studied in Press and Dyson (2012); Hilbe et al. (2013a, 2014, 2013b); Stewart and Plotkin (2013, 2014), our strategies are deterministic ones, which makes each of them easy to implement as a public policy without any randomization device (Dror, 1983). In particular, we focus on one of the friendly rivals, named CAPRI, because it can be described in plain language (Sec. 3), which implies great potential importance in understanding and guiding human behaviour. We also argue that our friendly rivals exhibit evolutionary robustness in the sense of Stewart and Plotkin (2013) for any population size and for any benefit-to-cost ratio. This property is demonstrated by evolutionary simulation in which CAPRI overwhelms other strategies under a variety of environmental parameters.

2. Search for friendly rivals

Despite the fundamental importance of memory in direct reciprocity, combinatorial explosion has been a major obstacle in understanding the memory effects on cooperation: Let us consider deterministic strategies with memory length $m$, which means that each of them chooses an action between $c$ and
Figure 1: (Left) A schematic diagram of the strategy space. Strategies that tend to cooperate (defect) are shown on the left (right). The blue ellipse represents a set of efficient strategies, which are cooperative to sustain mutual cooperation. On the other hand, the red ellipse represents a set of defensible strategies, which often defect to defend themselves from malicious co-players. In general, their intersection is small. When \( m = 2 \), for instance, the sizes of efficient and defensible sets are 7639 and 2144, respectively, whereas the intersection contains only eight strategies. (Right) The diamond depicts the region of possible average payoffs for Alice and Bob. The blue triangle shows the feasibility region when Alice uses a defensible strategy. If Alice and Bob both use the same strategy satisfying efficiency, they will reach \((R, R)\) (the blue dot).

\( d \) as a function of the \( m \) previous rounds. The number of such memory-\( m \) strategies expands as \( N = 2^{2^m} \), which means \( N_{m=1} = 16 \), \( N_{m=2} = 65536 \), and \( N_{m=3} \approx 1.84 \times 10^{19} \). The number of combinations of these strategies grows even more drastically, which renders typical evolutionary simulation incapable of exploring the full strategy space. Here, we take an axiomatic approach as in \( \text{Yi et al.} \) (2017); \( \text{Hilbe et al.} \) (2017); \( \text{Murase and Baek} \) (2018) to find friendly rivals. That is, we search for strategies that satisfy certain predetermined criteria, and the computation time for checking those criteria scales as \( O(N) \) instead of \( O(N^2) \) or greater.

More specifically, we begin with the following two criteria (\( \text{Yi et al.} \) 2017; \( \text{Murase and Baek} \) 2018):

1. Efficiency: Mutual cooperation is achieved with probability one as error probability \( e \to 0^+ \), if both Alice and Bob use this strategy.

2. Defensibility: If Alice uses this strategy, she will never be outperformed by Bob when \( e = 0 \) regardless of initial actions. This is a sufficient condition for being a rival, i.e., \( \lim_{e \to 0^+} (\pi_A - \pi_B) \geq 0 \).

The efficiency criterion requires a strategy to establish cooperation in the presence of small \( e \) when both the players adopt this strategy. This criterion is satisfied by many generous strategies such as unconditional cooperation.
(AllC), generous TFT (GTFT), Win-Stay-Lose-Shift (WSLS) and Tit-for-two-tats (TF2T). Partner strategies constitute a sub-class of efficient ones by limiting the co-player’s payoff to be less than or equal to $R$ regardless of the co-player’s payoff \( [Akin, 2016; Hilbe et al., 2015, 2018a] \). On the other hand, a defensible strategy must ensure that the player’s long-time average payoff will be no less than that of the co-player who may use any possible strategy, and this idea is equivalent to the notion of a ‘rival strategy’ \( [Hilbe et al., 2015, 2018a] \). Defensible strategies include unconditional defection (AllD), TFT, and extortionate ZD strategies. Figure 1(a) schematically shows how these two criteria narrow down the list of strategies to consider. The overlap of efficient and defensible strategies means a set of friendly rivals because it is a subset of partner strategies. It assigns the most strict limitation on the co-player’s payoff among the partner strategies as shown in Fig. 1(b).

Indeed, the overlap region between these two criteria is extremely tiny: It is pure impossibility for $m = 1$, and we find only 8 strategies out of $N = 65536$ for $m = 2$.

To further narrow down the list of strategies, we impose the third criterion \( [Yi et al., 2017; Murase and Baek, 2018] \):

3. Distinguishability: The strategy has a strictly higher payoff than the co-player’s when its strategy is AllC in small error limit, i.e., \( \lim_{e \to 0^+} (\pi_A > \pi_{AllC}) \).

This requirement originates from evolutionary game theory: If this criterion is violated, the number of AllC players may increase due to neutral drift, which eventually makes the population vulnerable to invasion of defectors such as AllD. We check these the criteria for each strategy by representing it as a graph and analysing its topological properties (See Appendix for details). If a strategy satisfies all those three criteria, it will be called ‘successful’.

Among deterministic memory-two strategies, it is known that only four strategies satisfy these three conditions \( [Yi et al., 2017] \). They have minor differences from each other, and one of them is called TFT-ATFT, which is a combination of TFT and anti-tit-for-tat (ATFT). It usually behaves as TFT, but it takes the opposite moves after mistakenly defecting from mutual cooperation. Similar analysis has been conducted for the three-person public-goods game: At least 256 successful strategies exist when $m = 3$, whereas no such solution exists when $m < 3$ \( [Murase and Baek, 2018] \). It has also been shown that a friendly rival strategy must have $m \geq n$ for the general $n$-person public-goods game, although such a strategy for $n > 3$ is yet to be
found. These results suggest that a novel class of strategies may appear as the memory length exceeds a certain threshold.

For memory-three strategies, we have obtained an exhaustive list of successful strategies by massive supercomputing (see Appendix A for detail). The efficiency and defensibility criteria find 7,018,265,885,034 friendly rivals out of $N_{m=3} = 2^{64} \approx 1.84 \times 10^{19}$ strategies. If the distinguishability criterion is additionally imposed, 4,261,844,305,281 strategies are found. There are four actions commonly prescribed by all these successful strategies: Let $A_t$ and $B_t$ denote Alice’s and Bob’s actions at time $t$, respectively. When their memory states are $(A_{t-3}, A_{t-2}, A_{t-1}, B_{t-3}, B_{t-2}, B_{t-1}) = (ccc, ccc), (ccc, ddd), (cdd, ddd)$, and $(ddd, ddd)$, all the successful strategies tell Alice to choose $c, d, d$, and $d$, respectively. The first one is absolutely required to maintain mutual cooperation. The latter three are needed to satisfy the defensibility criterion: If $c$ was prescribed at any of these states, Alice would be exploited by Bob’s continual defection.

Table 1: Recovery paths to mutual cooperation for the memory-three successful strategies. Only the most common five patterns are shown in this table. The upper and lower rows represent the sequences of actions taken by Alice and Bob, respectively, when Bob defected from mutual cooperation by error. The right column shows the number of strategies having each pattern, as well as its fraction with respect to the total number of successful strategies.

| action sequence       | # of strategies         |
|-----------------------|-------------------------|
| $... \ cd\ cd\ c\ ...$ | 905,772,524,235 $(21.3\%)$ |
| $... \ cd\ cd\ c\ ...$ |                         |
| $... \ cd\ cd\ d\ c\ ...$ | 522,061,013,252 $(12.2\%)$ |
| $... \ cd\ cd\ c\ ...$ |                         |
| $... \ cd\ cd\ c\ ...$ | 437,671,509,356 $(10.3\%)$ |
| $... \ cd\ cd\ d\ c\ ...$ |                         |
| $... \ cd\ cd\ d\ c\ ...$ | 409,458,612,318 $(9.6\%)$ |
| $... \ cd\ cd\ d\ c\ ...$ |                         |
| $... \ cd\ cd\ d\ c\ ...$ | 227,113,898,468 $(5.3\%)$ |
| $... \ cd\ cd\ d\ c\ ...$ |                         |
| $... \ cd\ cd\ d\ c\ ...$ | 184,052,002,852 $(4.3\%)$ |

Except for these four prescriptions, we see a wide variety of patterns. For example, let us assume that both Alice and Bob adopt one of these strategies.
When Bob defects by error, they must follow a recovery path from state (ccc, ccd) to (ccc, ccc). We find 839 different patterns from our successful strategies (Table 1). The most common one is also the shortest, in which only two time steps are needed to recover mutual cooperation. It cannot be shorter because Alice must defect at least once to assure defensibility. It is even shorter than that of TFT-ATFT, which is identical to the third entry of Table 1. This finding disproves the speculation in Yi et al. (2017) that friendly rivals are limited to a variant of TFT even if $m > 2$ (see also Appendix B for other examples). This shortest recovery path is possible only when $m \geq 3$, indicating a pivotal role of memory length in direct reciprocity.

3. CAPRI strategy

The shortest recovery path in Table 1 shows that Bob can recover his own mistake simply by accepting Alice’s punishment provided that he has $m = 3$. Among the strategy using this recovery pattern, we have discovered a strategy which is easy to interpret, named ‘CAPRI’, after the first letters of its five constitutive rules listed below:

1. Cooperate at mutual cooperation.
   - $(ccc, ccc) \rightarrow c$

2. Accept punishment when you mistakenly defected from mutual cooperation.
   - $(ccd, ccc) \rightarrow c$
   - $(cde, ccd) \rightarrow c$
   - $(ddd, cde) \rightarrow c$
   - $(ccc, dcd) \rightarrow c$

3. Punish your co-player by defecting once when he defected from mutual cooperation.
   - $(ccc, ccd) \rightarrow d$
   - $(cdd, cdc) \rightarrow c$
   - $(cde, dcc) \rightarrow c$
   - $(dcd, ccc) \rightarrow c$
4. Recover cooperation when you or your co-player cooperated at mutual defection.

- \((dd, dd, dc) \rightarrow c\)
- \((dc, ddc) \rightarrow c\)
- \((ddc, ccc) \rightarrow c\) (duplicate of rule 3)
- \((ddc, ddd) \rightarrow c\)
- \((dcc, ddc) \rightarrow c\)
- \((ccc, ddc) \rightarrow c\) (duplicate of rule 2)
- \((ddc, ddc) \rightarrow c\)

5. In all the other cases, defect.

The first rule is clearly needed for efficiency. In addition, mutual cooperation must be robust against one-bit error, i.e., occurring with probability of \(O(e)\), when both Alice and Bob use this strategy. This property is provided by the second and the third rules. In addition, for this strategy to be efficient, the players must be able to escape from mutual defection through one-bit error so that the stationary probability distribution does not accumulate at mutual defection, which is handled by the fourth rule. Note that these four rules for efficiency do not necessarily violate defensibility when \(m > 2\), as we have already seen in Table 1. Actually, due to the fifth rule, both efficiency and defensibility are satisfied by CAPRI. The action table and its minimized automaton representation (Murase and Baek, 2019) are given in Table 2 and Fig. 2, respectively. The self-loop via \(dc\) at state ‘2’ in Fig. 2a proves that this strategy also satisfies distinguishability.

CAPRI requires memory three otherwise it violates defensibility. If CAPRI were a memory-two strategy, \((cd, dc) \rightarrow c\) and \((dc, cd) \rightarrow c\) must be prescribed to recover from error. However, with these subscription, Bob can repeatedly exploit Alice acting like the following:

\[
\begin{align*}
\ldots & c c d c c c c \ldots \\
\ldots & c d c d c \ldots
\end{align*}
\]  

TFT-ATFT and its variants are the only friendly rivals when memory length is less than three.
Compared with TFT-ATFT, CAPRI is closer to Grim trigger (GT) rather than to TFT. Alice keeps cooperating as long as Bob cooperates, but she switches to defection, as prescribed by the fifth rule, when Bob does not conform to her expectation. Due to the similarity of CAPRI to GT, it also outperforms a wider spectrum of strategies than TFT-ATFT. Figure 2(b) shows the distribution of payoffs of the two players when Alice’s strategy is CAPRI, and Bob’s strategy is sampled from the 64-dimensional unit hypercube of memory-three probabilistic strategies. Alice’s payoff is strictly higher than Bob’s in most of the samples. On the other hand, when Alice uses TFT-ATFT, payoffs are mostly sampled on the diagonal because it is based on TFT, which equalizes the players’ payoffs. However, we also note that CAPRI is significantly different from GT in two ways. First, CAPRI is error-tolerant: Even when Bob makes a mistake, Alice is ready to recover cooperation after Bob accepts punishment, as described in the second and the third rules. Second, whereas GT is characterized by its irreversibility, CAPRI let the players escape from mutual defection according to the fourth rule.

Table 2: Action table of CAPRI. The superscript on the upper left corner of each element indicates which rule is involved.

| $A_{t-3}A_{t-2}A_{t-1}$ | $B_{t-3}B_{t-2}B_{t-1}$ |
|-------------------------|------------------------|
| ccc                     | 1, 3, d, d, d, 2, 4, c, d, d, d, d |
| ccd                     | 2, 3, c, d, d, d, d, d, d, d |
| cdc                     | d, 2, c, d, d, d, d, d, d, d |
| cdd                     | 3, 4, c, d, 2, c, d, 4, c, d, 4, c, d |
| dcd                     | d, d, d, d, d, d, d, d, d, d |
| ddc                     | d, d, d, d, 4, c, d, 4, c, 4, c, d |
| ddd                     | d, d, d, d, d, d, 4, c, d, 4, c, d |

4. Evolutionary simulation

Although defensibility assures that the player is never outperformed by the co-player, it does not necessarily guarantee success in evolutionary games, where everyone is pitted against every other in the population. For example,
extortionate ZD strategies perform poorly in an evolutionary game (Stewart and Plotkin, 2013; Hilbe et al., 2013b; Adami and Hintze, 2013). In this section, we will check the performance of CAPRI in the evolutionary context.

When we consider performance of a strategy in an evolving population, the most famous measure of assessment is evolutionary stability (ES) (Maynard Smith, 1982). Although conceptually useful, ES is a too strong condition, requiring that when a sufficient majority of population members apply the strategy, every other strategy is at a selective disadvantage. Stewart and Plotkin (2013) introduced a more practical notion of stability: A strategy is called evolutionary robust if no other strategy has fixation probability greater than $1/N$, which is the fixation probability of a neutral mutant. In other words, an evolutionary robust strategy cannot be selectively replaced by any mutant strategy. Evolutionary robustness of a strategy depends on the population size: Partner strategies have this property when $N$ is large enough, whereas for rival strategies, it is when $N$ is small (Stewart and Plotkin, 2013). Friendly rivals have the virtue of both: They keep evolutionary robustness regardless of $N$, as will be shown in the following.

As in a standard stochastic model proposed by Imhof and Nowak (2010), let us consider a well-mixed population of size $N$ in which selection follows an imitation process. At each discrete time step, a pair of players are chosen at random, and we will call their strategies $X$ and $Y$, respectively. The
Figure 3: (a) Abundance of memory-one partners, rivals, and the other strategies. We consider a simplified version of the PD game, parametrized by \( b \) and \( c \), where \( b \) is the benefit to the co-player and \( c \equiv 1 \) is the cost of cooperation. In terms of the elementary payoffs, this corresponds to \( R = b - c \), \( T = b \), \( S = -c \), and \( P = 0 \). The Moran process is simulated with selection strength \( \sigma \) in a population of size \( N \), where the product \( N\sigma \) is fixed as 10. Following [Hilbe et al. (2018a)], three parameters (benefit-cost-ratio \( b \), population size \( N \), and error rate \( e \)) are varied one by one. Their default values are \( b = 3 \), \( N = 50 \), \( e = 10^{-3} \) unless otherwise stated. We also show the simulation results with (b) TFT-ATFT, (c) CAPRI, and (d) both TFT-ATFT and CAPRI, introduced with probability \( \mu = 0.01 \). These are average results over 10 Monte-Carlo runs.
probability for one of the players to replace her strategy $X$ with $Y$ is given as follows:

$$f_{x\rightarrow y} = \frac{1}{1 + \exp[\sigma (s_x - s_y)],} \quad (3)$$

where $s_x$ and $s_y$ denote the average payoffs of $X$ and $Y$ against the entire population, respectively, and $\sigma$ is a parameter which denotes the strength of selection. In population dynamics, we assume that the mutation rate $\mu$ is low enough: That is, when a mutant strategy $X$ appears in a resident population of $Y$, no other mutant will be introduced until $X$ reaches fixation or goes extinct. The dynamics is formulated as a Moran process, under which the fixation probability of $X$ is given in a closed form (Stewart and Plotkin, 2013):

$$\rho = \frac{1}{\sum_{i=0}^{N-1} \prod_{j=1}^{i} e^{\sigma([N-j-1]s_{yy} + js_{yx} - (N-j)s_{xy} - (j-1)s_{xx})}}, \quad (4)$$

where $s_{xy}$ denotes the long-term payoff of player $X$ against player $Y$. Using Jensen’s inequality, we see that

$$\frac{1}{\rho} = \sum_{i=0}^{N-1} e^{\sigma((2N-i-3)s_{yy} + (i+1)s_{yx} - (2N-i-1)s_{xy} - (i-1)s_{xx})/2} \geq Ne^{\sigma(N-1)((N-2)s_{yy} - s_{xx}) + (N+1)(s_{yx} - s_{xy}) + (N-2)(s_{yy} - s_{xy})}/6. \quad (5)$$

When $Y$ is a partner strategy, it satisfies $s_{yy} \geq s_{xy}$ and $s_{yy} \geq s_{xx}$. When $Y$ is also a rival strategy, it has another inequality, $s_{yx} \geq s_{xy}$. Therefore, the fixation probability of an arbitrary mutant $\rho \leq 1/N$ regardless of $N$ and $\sigma$.

We have conducted evolutionary simulation to assess the performance of friendly rivals. First, we run simulation without CAPRI and TFT-ATFT. This simulation adopts the setting used in Hilbe et al. (2018a) and serves as a baseline of performance. A mutant strategy is restricted to reactive memory-one strategies, according to which the player’s action depends only on the co-player’s last action. The reactive strategies are characterized by a pair of probabilities $(p_c, p_d)$, where $p_\alpha$ denotes the probability to cooperate when the co-player’s last move was $\alpha$. Rival strategies are represented by $p_d = 0$, and partners are by $p_c = 1$ and $p_d < p_d^*$, where $p_d^* \equiv \min\{1 - (T - R)/(R - S), (R - P)/(T - P)\}$. Mutant strategies may be randomly drawn from $[0, 1] \times [0, 1]$, but we have discretized the unit square in a way that each $p_\alpha$ takes a value from $[0/s, 1/s, \ldots, s/s]$ with $s = 10$. We have run the simulation until mutants are introduced $10^7$ times, and measured how frequently partner or
rival strategies are observed. As shown in Fig. 3(a), evolutionary performance of strategies depends on environmental parameters (Stewart and Plotkin, 2013, 2014; Hilbe et al., 2018a). Rival strategies have higher abundance when the benefit-to-cost ratio is low, population size $N$ is small, and error rate $e$ is high. Otherwise, partner strategies are favoured.

Let us now assume that a mutant can also take TFT-ATFT in addition to the reactive memory-one strategies. Figure 3(b) shows that TFT-ATFT occupies significant fractions across a broad range of parameters. The situation changes even more remarkably when CAPRI is introduced instead of TFT-ATFT. As seen in Fig. 3(c), CAPRI overwhelms the other strategies for almost the entire parameter ranges. The low abundance at $N = 2$ or $e = 10^{-5}$ does not contradict with the evolutionary robustness of CAPRI because it is still higher than the abundance of a neutral mutant. Furthermore, by comparing Figs. 3(b) and 3(c), we see that CAPRI shows better performance than TFT-ATFT. The evolutionary advantage of CAPRI over TFT-ATFT is directly observed in Fig. 3(d), where both CAPRI and TFT-ATFT are introduced into the population. As we have seen in Fig. 2(b), it tends to earn strictly higher payoffs against various types of co-players, whereas TFT-ATFT, based on TFT, aims to equalize the payoffs except when it encounters naive cooperators. This observation shows a considerable amount of diversity even among evolutionary robust strategies, which is in agreement with Stewart and Plotkin (2016).

5. Summary and discussion

To summarize, we have investigated the possibility to act as both a partner and a rival in the repeated PD game without future discounting. By thoroughly exploring a huge number of strategies with $m = 3$, we have found that it is indeed possible in various ways. The resulting friendly rivalry directly implies evolutionary robustness for any population size, benefit-to-cost ratio, and selection strength. We observe its success even when $e$ is of a considerable size (Fig. 3). It is also worth noting that a friendly rival can publicly announce its strategy because it is guaranteed not to be outperformed regardless of the co-player’s prior knowledge. Rather, it is desirable that the strategy should be made public because the co-player can be advised to adopt the same strategy by knowing it from the beginning to maximize its payoff. The resulting mutual cooperation is a Nash equilibrium. The deterministic nature offers additional advantages because the player can
implement the strategy without any randomization device. Moreover, even if uncertainty exists in the cost and benefit of cooperation, a friendly rival retains its power because it is independent of \((R, T, S, P)\). This is a distinct feature compared to the ZD strategies, whose cooperation probabilities have to be calculated from the elementary payoffs. Furthermore, the results are independent of the specific payoff ordering \(T > R > P > S\) of the PD. These are valid as long as mutual cooperation is socially optimal \((R > P\) and \(2R > T + S\)) and exploiting the other’s cooperation pays better than being exploited \((T > S)\). This condition includes other well-known social dilemma, such as the snowdrift game (with \(T > R > S > P\)) and the stag-hunt game (with \(R > T > P > S\)).

This work has focused on one of friendly rivals, named CAPRI. We speculate that it is close to the optimal one in several respects: First, it recovers mutual cooperation from erroneous defection in the shortest time. Second, it outperforms a wide range of strategies. Furthermore, its simplicity is almost unparalleled among friendly rivals discovered in this study. CAPRI is explained by a handful of intuitively plausible rules (Sec. 3), and such simplicity greatly enhances its practical applicability because the required cognitive load will be low when we humans play the strategy (Wedekind and Milinski, 1996; Milinski and Wedekind, 1998; Hilbe et al., 2014). It is an interesting research question whether this statement can be verified experimentally.

In particular, we would like to stress the importance of memory length in theory and experiment, considering that much research attention has been paid to the study of memory-one strategies (Stewart and Plotkin, 2013, 2014; Akin, 2016; Hilbe et al., 2015; Back and Kim, 2008; Hilbe et al., 2018b; Ichniwo and Masuda, 2018). Besides the combinatorial explosion of strategic possibilities, one can argue that a memory-one strategy, if properly designed, can unilaterally control the co-player’s payoff even when the co-player has longer memory (Press and Dyson, 2012). It has also been shown that \(m = 1\) is enough for evolutionary robustness against mutants with longer memory (Stewart and Plotkin, 2013). However, as pointed out in Stewart and Plotkin (2013, 2016), the payoff that a strategy receives against itself may depend on its own memory capacity, and this is the reason that a friendly rival is feasible when \(m > 1\). We can gain some important strategic insight only by moving beyond \(m = 1\).

In a broader context, although ‘friendly rivalry’ sounds self-contradictory, the term captures a crucial aspect of social interaction when it goes in a productive way: Rivalry is certainly ubiquitous between artists, sports teams,
firms, research groups, or neighbouring countries (Hogan, 2007; Brandenburger and Nalebuff, 2011; Kilduff, 2014; Pike et al., 2018). At the same time, they are subject to repeated interaction, whereby they eventually become friends, colleagues, or business partners to each other. Our finding suggests that such a seemingly unstable relationship can readily be sustained just by following a few simple rules: Cooperate if everyone does, accept punishment for your mistake, punish defection, recover cooperation if you find a chance, but in all the other cases, just take care of yourself. These seem to be the constituent elements for such a sophisticated compound of rivalry and partnership.

Acknowledgements

The authors would like to thank C. Hilbe for his careful reading and insightful comments on the manuscript. Y.M. acknowledges support from MEXT as “Exploratory Challenges on Post-K computer (Studies of multi-level spatiotemporal simulation of socioeconomic phenomena)” and from Japan Society for the Promotion of Science (JSPS) (JSPS KAKENHI; Grant no. 18H03621). S.K.B. acknowledges support by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (Grant No. NRF-2017R1A1A1A05001482). The source code for this study is available at https://github.com/yohm/sim_exhaustive_m3_PDgame. This research used computational resources of the K computer provided by the RIKEN Center for Computational Science through the HPCI System Research project (Project ID:hp160264). OACIS was used for the simulations in this study (Murase et al., 2018).

References

Adami, C., Hintze, A., 2013. Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. Nature communications 4 (1), 1–8.

Akin, E., 2015. What you gotta know to play good in the iterated Prisoners Dilemma. Games 6 (3), 175–190.

Akin, E., 2016. The Iterated Prisoner’s dilemma: good strategies and their dynamics. In: Assani, I. (Ed.), Ergodic Theory, Advances in Dynamical Systems. de Gruyter, Berlin, pp. 77–107.
Baek, S. K., Kim, B. J., 2008. Intelligent Tit-for-Tat in the iterated prisoner’s dilemma game. Phys. Rev. E 78 (1), 011125.

Brandenburger, A. M., Nalebuff, B. J., 2011. Co-opetition. Currency Doubleday, New York.

Dror, Y., 1983. Public Policymaking Reexamined. Transaction Publishers, New Brunswick.

Duersch, P., Oechssler, J., Schipper, B. C., 2012. Unbeatable imitation. Games Econ. Behav. 76 (1), 88–96.

Hilbe, C., Chatterjee, K., Nowak, M. A., 2018a. Partners and rivals in direct reciprocity. Nat. Hum. Behav. 2 (7), 469–477.

Hilbe, C., Martinez-Vaquero, L. A., Chatterjee, K., Nowak, M. A., 2017. Memory-n strategies of direct reciprocity. Proc. Natl. Acad. Sci. USA 114 (18), 4715–4720.

Hilbe, C., Nowak, M. A., Sigmund, K., 2013a. Evolution of extortion in Iterated Prisoner’s Dilemma games. Proc. Natl. Acad. Sci. USA 110 (17), 6913–6918.

Hilbe, C., Nowak, M. A., Traulsen, A., 2013b. Adaptive dynamics of extortion and compliance. PloS one 8 (11), e77886.

Hilbe, C., Schmid, L., Tkadlec, J., Chatterjee, K., Nowak, M. A., 2018b. Indirect reciprocity with private, noisy, and incomplete information. Proc. Natl. Acad. Sci. USA 115 (48), 12241–12246.

Hilbe, C., Traulsen, A., Sigmund, K., 2015. Partners or rivals? Strategies for the iterated prisoner’s dilemma. Games Econ. Behav. 92, 41–52.

Hilbe, C., Wu, B., Traulsen, A., Nowak, M. A., 2014. Cooperation and control in multiplayer social dilemmas. Proc. Natl. Acad. Sci. USA 111 (46), 16425–16430.

Hogan, J., 2007. Behind the hunt for the Higgs boson. Nature 445, 239.

Hougardy, S., 2010. The Floyd–Warshall algorithm on graphs with negative cycles. Inf. Process. Lett. 110 (8-9), 279–281.
Ichinose, G., Masuda, N., 2018. Zero-determinant strategies in finitely repeated games. J. Theor. Biol. 438, 61–77.

Imhof, L. A., Nowak, M. A., 2010. Stochastic evolutionary dynamics of direct reciprocity. Proc. R. Roc. B 277 (1680), 463–468.

Kilduff, G. J., 2014. Driven to win: Rivalry, motivation, and performance. Soc. Psychol. Pers. Sci. 5 (8), 944–952.

Maynard Smith, J., 1982. Evolution and the Theory of Games. Cambridge Univ. Press.

Milinski, M., Wedekind, C., 1998. Working memory constrains human cooperation in the Prisoners Dilemma. Proc. Natl. Acad. Sci. USA 95 (23), 13755–13758.

Morone, F., Min, B., Bo, L., Mari, R., Makse, H. A., 2016. Collective influence algorithm to find influencers via optimal percolation in massively large social media. Sci. Rep. 6, 30062.

Murase, Y., Baek, S. K., 2018. Seven rules to avoid the tragedy of the commons. J. Theor. Biol. 449, 94–102.

Murase, Y., Baek, S. K., 2019. Automata representation of successful strategies for social dilemmas. arXiv:1910.02634.

Murase, Y., Uchitane, T., Ito, N., 2018. An open-source job management framework for parameter-space exploration: Oacis. arXiv preprint arXiv:1805.00438.

Nowak, M. A., 2006. Five rules for the evolution of cooperation. Science 314 (5805), 1560–1563.

Pike, B. E., Kilduff, G. J., Galinsky, A. D., 2018. The long shadow of rivalry: Rivalry motivates performance today and tomorrow. Psychol. Sci. 29 (5), 804–813.

Press, W. H., Dyson, F. J., 2012. Iterated Prisoner’s Dilemma contains strategies that dominate any evolutionary opponent. Proc. Natl. Acad. Sci. USA 109 (26), 10409–10413.
Appendix A. Method

We wish to examine the strategy space of $m = 3$, but it is impossible to enumerate all the memory-three strategies by a naive brute-force method even if we use a cutting-edge supercomputer because their total number is as large as $2^{2^m} = 2^{64} \approx 2 \times 10^{19}$. To overcome this difficulty, we have developed graph-theoretic algorithms to judge defensibility, efficiency, and distinguishability. In the following, we explain the algorithm in three steps: First, we present basic ideas to judge the three criteria for a single strategy. Second, we show how this can be done for a set of strategies simultaneously. Third, we apply these algorithms to enumerate all successful strategies comprehensively in the memory-three strategy space. A C++ source code is available under an open-source license at https://github.com/yohm/sim_exhaustive_m3_PDgame.

Appendix A.1. Judging the criteria for a single strategy

Let us consider two players $A$ and $B$ in the iterated PD game. Player $A$’s action at time $t$ is denoted as $A_t$, and $B_t$ is defined likewise. When $m = 3$, we have 64 different history profiles, $(A_{t-3}A_{t-2}A_{t-1}, B_{t-3}B_{t-2}B_{t-1}) = (ccc,ccc)$,
(ccc, ccd), (ccc, cdc), ... (ddd, ddd). These profiles can also be represented as 0, 1, 2, ... 63 in binary. Consider a directed graph whose nodes represent the history profiles and whose links represent transition among them as prescribed by $S_A$ and $S_B$, where $S_A$ and $S_B$ are the strategies of the players $A$ and $B$, respectively. Such a graph will be called a transition graph in general. Due to the deterministic property of $S_A$ and $S_B$, each node has one outgoing link in the absence of error, so the total number of links is also 64. We will denote this graph as $g(S_A, S_B)$.

We may also consider another transition graph for the case where $B$’s actions are left undetermined whereas $A$’s strategy is $S$, namely $g(S, *)$. Player $B$ may choose either $c$ or $d$, thus each node has two outgoing links. This graph is useful in judging the defensibility of $S$: This criterion concerns relative payoff differences, which are made by either unilateral cooperation or unilateral defection. Traversing every possible cycle in $g(S, *)$, therefore, we count the number of nodes with $(A_{t-1}, B_{t-1}) = (c, d)$ and subtract it from the number of nodes with $(A_{t-1}, B_{t-1}) = (d, c)$. Working with integer counts is also numerically convenient. If none of the cycles in $g(S, *)$ gives a negative value in this counting, we can say that the strategy $S$ satisfies the defensibility criterion.

A conventional way to judge the efficiency criterion of $S$ is to consider a transition graph of $S$ against itself, but with error probability $e$. Due to error, both the players can choose either $c$ or $d$, which means that each node of the transition graph has four outgoing links. One can check the corresponding stationary probability $\vec{\pi} = (\pi_0, \ldots, \pi_{63})^\top$, where $\top$ means transpose. The strategy $S$ is efficient if $\pi_0$ converges to 100% as the error probability $e$ approaches zero from above. The calculation of $\vec{\pi}$ can be done through linear algebraic calculation with decreasing $e$ gradually, as has been done in [Yi et al. (2017)] or [Murase and Baek (2018)].

The above method takes into account the effects of error all at once. However, we can devise a quicker way to judge the efficiency criterion topologically with increasing the order of $e$ one by one, as we will explain now. Let us begin with $g_0 = g(S, S)$ which does not take into account any error. It is a directed graph, either connected or disconnected. We write $i \rightarrow j$ if node $j$ is reachable from $i$ in $g_0$, and $i \not\rightarrow j$ otherwise. When they are mutually reachable (unreachable), we write $i \leftrightarrow j$ ($i \not\leftrightarrow j$). If a node has no outgoing links, it is called a sink. This notion is extended to a strongly connected component (SCC) as well, that is, a SCC is also called a sink if it has no outgoing links. If $\alpha$ is a SCC composed of nodes $\alpha_1, \alpha_2, \ldots, \alpha_s$, the
stationary distribution over $\alpha$ is defined as $\pi_\alpha = \pi_{\alpha_1} + \pi_{\alpha_2} + \ldots + \pi_{\alpha_s}$. If a sink is reachable from a node, we say that the node is in the basin of the sink, where the basin includes the sink itself.

If $i \rightarrow 0$ and $0 \not\rightarrow i$ for every node $i \neq 0$ in $g_0$, the node 0 constitutes the unique sink of this graph: It is a sink, by definition. It is also unique because none of the other nodes is a sink. If this is the case, just by checking $g_0$ without considering error, we may conclude that the strategy $S$ under consideration satisfies the efficiency criterion. Although we have not taken into account error-induced transitions, this conclusion can be justified in two ways: First, the detailed-balance condition implies that $\pi_i/\pi_0 \lesssim O(e)$ for every $i \neq 0$ because $i$ can be accessed from the sink 0 only by error. Or, we can explicitly construct the derivative of the principal eigenvector by using the fact that it is non-degenerate \cite{van der Aa et al. 2007}, which implies that error-induced change in connectivity with a size of $e \ll 1$ perturbs the stationary distribution $\vec{\pi}$ by an amount of $O(e)$ at most. As $e \rightarrow 0^+$, therefore, $\pi_0$ will approach 100%.

However, if the above condition is not met, i.e., if multiple sinks coexist, it is impossible to judge the efficiency of $S$ from $g_0$. We then need to take error-induced transitions into consideration. For example, let us consider $g_0$ with two sinks $\alpha$ and $\beta$, together with their respective basins $\Omega_\alpha(0)$ and $\Omega_\beta(0)$. In other words, every node $i$ in this graph satisfies $i \rightarrow \alpha$ or $i \rightarrow \beta$. We define $\rho_\alpha^{(1)}$ as a set of nodes that are reachable from $\alpha$ via a single error, and define $\rho_\beta^{(1)}$ likewise. Formally speaking, we define $\rho_\alpha^{(k)} \equiv \{i|\alpha \rightarrow_k i\}$, where the subscript $k$ below the arrow means that we are considering transitions that are mediated by $k$ errors at least. The simple reachability relation $\alpha \rightarrow i$ is equivalent to $\alpha \rightarrow_0 i$. We will assume that $\rho_\beta^{(1)}$ has an overlap with $\Omega_\alpha(0)$ at node $j$, whereas $\rho_\alpha^{(1)}$ does not with $\Omega_\beta(0)$. It means that $\beta \rightarrow j \rightarrow \alpha$ whereas $\alpha \not\rightarrow \beta$. Then, for every node $i$ in $\Omega_\beta(0)$, we find the following path: $i \rightarrow \beta \rightarrow j \rightarrow \alpha$. Strictly speaking, $\alpha$ is no longer a sink at this level of description because $\alpha \rightarrow_1 l \in \rho_\alpha^{(1)}$ by the definition of $\rho_\alpha^{(1)}$. However, we expect that such transitions should not alter the situation significantly because $\rho_\alpha^{(1)}$ is still a subset of $\Omega_\alpha(0)$, meaning that the dominant direction is $i \rightarrow \alpha$ with probability of $O(1)$. Furthermore, once the principal eigenvector $\vec{\pi}$ becomes non-degenerate due to the error-induced transition via $\rho_\beta^{(1)}$, all other perturbations $\lesssim O(e)$ that we have neglected add to $\vec{\pi}$ only small
changes which vanish in the small-$\varepsilon$ limit, as can be seen from the fact that the principal eigenvector has a well-defined derivative (van der Aa et al., 2007). To summarize, this double-sink example has the following property:

\[
\begin{cases}
  i \to \alpha \text{ and } \alpha \not\to i & \text{if } i \in \Omega^{(0)}_{\alpha} \\
  \not\leftrightarrow & \text{if } i \in \Omega_{\alpha}^{(1)} \\
  \not\to 1 & \text{if } i \in \Omega_{\beta}^{(0)}
\end{cases}
\] (A.1)

where $\Omega^{(0)}_{\alpha} \cup \Omega^{(0)}_{\beta}$ equals the whole set of nodes by assumption, and Eq. (A.1) guarantees that $\pi_\alpha$ approaches 100% in the limit of $\varepsilon \to 0^+$. In other words, the original basin $\Omega^{(0)}_{\alpha}$ has been extended to $\Omega^{(1)}_{\alpha} \equiv \Omega^{(0)}_{\alpha} \cup \Omega^{(0)}_{\beta}$ in the sense of Eq. (A.1), and $\pi_\alpha$ approaches 100% as the new basin of $\alpha$ coincides with the whole set of nodes. Another way to rephrase it is to construct $g_1$ by supplementing $g_0$ with additional links from $\gamma$ to the member nodes of $\rho^{(1)}_{\gamma}$, where $\gamma \in \{\alpha, \beta\}$ denotes each sink of $g_0$. In terms of $g_1$, it can be said as follows: We have $\pi_\alpha \to 100\%$ in the small-$\varepsilon$ limit because every node $i$ in $g_1$ has a certain integer $k_i \in \{0, 1\}$ such that

\[
\begin{cases}
  i \not\leftrightarrow \alpha & \text{for } 0 \leq k < k_i \\
  i \to \alpha & \text{for } k = k_i
\end{cases}
\] (A.2)

As a more concrete example, let us consider a Markovian system with a set of four nodes, $\{\alpha, \alpha', \alpha'', \beta\}$. The transition matrix is given as follows:

\[
W = \begin{pmatrix}
1 - \varepsilon & 1 & 0 & 0 \\
\varepsilon & 0 & 0 & \varepsilon \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 - \varepsilon
\end{pmatrix}
\]

each of whose columns adds up to one. At $\varepsilon = 0$, the system has two disconnected parts, $\Omega^{(0)}_{\alpha} = \{\alpha, \alpha'\}$ and $\Omega^{(0)}_{\beta} = \{\alpha'', \beta\}$. The first part with the largest eigenvalue $\lambda_\alpha(\varepsilon = 0) = 1$ has the corresponding left and right eigenvectors, $\vec{y}_\alpha = (1, 1, 0, 0)$ and $\vec{x}_\alpha = (1, 0, 0, 0)^T$, respectively. Likewise, the second part with the largest eigenvalue $\lambda_\beta(\varepsilon = 0) = 1$ has $\vec{y}_\beta = (0, 0, 1, 1)$ and $\vec{x}_\beta = (0, 0, 0, 1)^T$. Note that $\vec{y}_\alpha \cdot \vec{x}_\alpha = \vec{y}_\beta \cdot \vec{x}_\beta = 1$ and $\vec{y}_\alpha \cdot \vec{x}_\beta = \vec{y}_\beta \cdot \vec{x}_\alpha = 0$. It is a usual practice to calculate eigenvalue perturbation (Morone et al., 2016),
but due to the two-fold degeneracy of this problem, we have to diagonalize the following $2 \times 2$ matrix:

$$L = \begin{pmatrix} \vec{y}_\alpha \cdot W' \cdot \vec{x}_\alpha & \vec{y}_\beta \cdot W' \cdot \vec{x}_\beta \\ \vec{y}_\beta \cdot W' \cdot \vec{x}_\alpha & \vec{y}_\beta \cdot W' \cdot \vec{x}_\beta \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} = PDP^{-1} \quad (A.5)$$

with

$$P = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \quad (A.6)$$

and

$$D = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}, \quad (A.7)$$

where the prime denotes differentiation with respect to $e$. The diagonal elements of $D$ imply that $\lambda_\alpha(e) = 1 + O(e^2)$ and $\lambda_\beta(e) = 1 - e + O(e^2)$ so that probability over $\Omega^{(0)}_\beta$ will eventually be absorbed into that over $\Omega^{(0)}_\alpha$ as soon as $e$ becomes positive. If we look at the structures of the left and right eigenvectors, their dot products with $W'$ in Eq. (A.5) clearly show that the important point is whether the error-induced transitions from a ‘sink’ go outside its basin. Although the first sink $\alpha$ survives in this example, the actual stationary distribution, $\vec{\pi} = (1/(1 + e), e/(1 + e), 0, 0)^T$, slightly differs from $\vec{x}_\alpha = (1, 0, 0, 0)^T$ because $\alpha \to \alpha'$ in Eq. (A.4). However, as we have already expected, the difference is insignificant in the sense that $\vec{\pi}$ converges to $\vec{x}_\alpha$ continuously as $e \to 0$. From $\pi_{\alpha''\gamma}/\pi_{\alpha\gamma} = 0$, we also note that transition $\alpha'' \to \alpha'$ eventually occurs with probability $e + (1 - e)e + (1 - e)^2e + \ldots = 1$ in the long run because of the self-loop at $\beta$. Although it is involved with an error with probability $e \ll 1$, the total probability in the long run may be of $O(1)$, and this is what affects the stationary distribution $\vec{\pi}$. In addition, as long as the relevant connections are all preserved as described in Eq. (A.2), the other error-induced transitions, which may actually be involved in calculating $\vec{\pi}$, do not change the conclusion: As an example, suppose that we add to Eq. (A.4) transition from every node to every other with probability $e^2$, represented by

$$(\delta W)_{ij} = \begin{cases} -3e^2 & \text{if } i = j \\ e^2 & \text{if } i \neq j. \end{cases} \quad (A.8)$$

The second example is to add a transition of probability $e$ from $\alpha'$ to $\alpha''$ by setting $W_{\alpha''\alpha'} = e$ and $W_{\alpha\alpha'} = 1 - e$ in Eq. (A.4). In both of these examples,
we can show by direct calculation that the resulting $\vec{\pi}$ keeps converging to $\vec{x}_\alpha$ in the small-$\epsilon$ limit. On the other hand, if we add a transition from $\alpha$ to $\alpha''$ with probability $\epsilon$, it extends $\rho^{(1)}_\alpha$ beyond $\Omega^{(0)}_\alpha = \{\alpha, \alpha'\}$, so neither $\alpha$ nor $\beta$ survives alone but they divide up the stationary distribution in the sense that $\pi_\alpha/\pi_\beta \sim O(1)$.

Now, the above procedure can be carried out recursively:

1. Construct $g_0$ with a set of nodes, $\mathcal{N}$.
   - If $0 \rightarrow i$ for a certain node $i$, the strategy is inefficient.
   - If $\Omega^{(0)}_0 \equiv \{i|i \rightarrow 0 \text{ and } 0 \not\rightarrow i\}$ equals $\mathcal{N}$, the strategy is efficient.
   - If $\Omega^{(0)}_0$ is a strict subset of $\mathcal{N}$, the efficiency criterion is undecidable from $g_0$. Go to the next step with $\nu = 1$.

2. Construct $g_\nu$ by adding links from every sink $\gamma$ surviving in $g_{\nu-1}$ to the member nodes of $\rho^{(\nu)}_\gamma$.
   - If $0 \rightarrow i$ for a node $i$ outside $\Omega^{(\nu-1)}_0$, the strategy is inefficient.
   - If $\Omega^{(\nu)}_0 \equiv \Omega^{(\nu-1)}_0 \cup \{i|i \rightarrow 0 \text{ and } 0 \not\rightarrow i\}$ equals $\mathcal{N}$, the strategy is efficient.
   - If $\Omega^{(\nu)}_0$ is a strict subset of $\mathcal{N}$, the efficiency criterion is undecidable from $g_\nu$. Go to the next step.

3. Increase $\nu$ by one, and go back to the previous step.

This algorithm always ends with a decision between ‘efficient’ and ‘inefficient’ because the graph becomes strongly connected if we include all possible types of error. A pseudo-code to judge efficiency is given in Fig. A.4.

The algorithm to judge the distinguishability criterion is similar to the one for the efficiency criterion as shown in the following. If $S$ is a distinguishable strategy, the stationary probability of state $\lim_{\epsilon \rightarrow 0} \pi_0 < 1$ when players of $S$ and AllC play the game. This is because an AllC player must have a smaller payoff than that of the co-player except at full cooperation. Although this was judged by a linear algebraic calculation in Yi et al. (2017), we can directly use the topological structure of a graph $g(S, \text{AllC})$ just as in the case of the efficiency criterion. To this end, we only have to construct graphs $g_\nu$ from $g_{\nu-1}$, where $g_0$ is $g(S, \text{AllC})$ (Fig. A.5).
```python
def is_efficient(strategy):
    judged = Array(64, false)
    judged[0] = true
    # => judged = [true, false, false, ..., false]

    # initialize g_n = g(S, S)
    gn = construct_g(strategy, strategy)

    until judged.all?:
        64.times do |i|
            next if judged[i]
            if gn.reachable(0, i) # 0->i
                return false # judged as inefficient
            end
            if gn.reachable(i, 0) # i->0 & 0!->i
                judged[i] = true
            end
        end
        gn = update_gn(gn) # g_n <- g_{n+1}
    end
    return true
end

def update_gn(g)
    g_new = g.clone
    sink_sccs = g.sink_strongly_connected_components
    sink_sccs.each do |sink|
        for_each_node_in(sink) do |n|
            # states reachable by an error from n.
            noised_states = [n^1, n^8]
            noised_states.each do |to|
                g_new.add_link(n, to) unless g_new.has_link?(n, to)
            end
        end
    end
    return g_new
end
```

Figure A.4: A pseudo-code to judge efficiency of a strategy.
```python
def is_distinguishable(strategy):
    judged = Array(64, false)
    judged[0] = true
    # => judged = [true, false, false, ..., false]

    # initialize g_n = g(S, AllC)
gn = construct_g(strategy, AllC)

    until judged.all?
        64.times do |i|
            next if judged[i]
            if gn.reachable(0, i) # 0->i
                return true # judged as distinguishable
            end
            if gn.reachable(i, 0) # i->0 && 0!->i
                judged[i] = true
            end
        end
        gn = update_gn(gn) # g_n <- g_{n+1}
    end
    return false
end
```

Figure A.5: A pseudo-code to judge distinguishability of a strategy. The method ‘update_gn’ is identical to the one in Fig. A.4.

![Strategy Tree Example](image)

Figure A.6: An example of a strategy tree of the memory-1 strategy space for the iterated PD game. A memory-1 strategy is represented by a binary string of length $2^{nm} = 4$, each of which corresponds to a leaf node of the strategy tree. The depth of the strategy tree is 4. Starting from the node, which corresponds to the strategy set whose actions are not determined at all, one of the actions is determined whenever we visit a child node. Internal nodes correspond to a strategy set. Examples of the transition graphs for strategy sets $g(S,*)$ are shown as well.
Appendix A.2. Exploring the memory-three strategy space

The space of memory-\( m \) strategies is represented by a complete binary tree of depth \( 2^m \). Figure A.6 shows a tree of memory-one strategies. In this representation, a leaf vertex (a vertex having no child vertices) corresponds to a specific strategy whereas an internal vertex (a vertex having child vertices) represents a set of strategies, a part of whose actions remain undetermined. The root vertex of the tree corresponds to the whole set of strategies in memory-\( m \) strategy space. Traversing the tree from the root to a leaf vertex by one step is equivalent to determining one of the undetermined actions. Hereafter, this sort of tree is called a strategy tree, and the set of strategies corresponding to an internal vertex is called a strategy set. The order in which actions are determined may be arbitrary in each subtree. For example, in Fig. A.6, the root vertex branches into two subtrees depending on what to do at \((c, c)\). If the answer is \( c \), we enter the left subtree, and the next question concerns what to do at \((c, d)\). If we choose \( d \) at \((c, c)\), on the other hand, we get into the right subtree, and what comes next is the choice at \((d, d)\).

We will begin by finding strategies that satisfy the efficiency and defensibility criteria. As will be explained below, we focus on necessary conditions for a strategy set to satisfy the efficiency or defensibility criteria: If the necessary conditions are violated at an internal vertex, the whole branch below it may be discarded without further consideration. For this reason, the computational cost crucially depends on in which order the actions are determined along the branches of the tree.

Appendix A.3. Checking the defensibility criterion for a strategy set

Let \( g(S, \ast) \) be the transition graph for a strategy set \( S \), which is defined as the largest common subgraph of \( g(S, \ast) \) for every member strategy \( S \in S \), where \( \ast \) is a wildcard character: If \( g(S, \ast) \) contains a negative cycle, \( g(S, \ast) \) must also contain it, hence \( S \) violates defensibility. Conversely, a necessary condition for \( S \) to satisfy the defensibility criterion is the absence of negative cycles in \( g(S, \ast) \). Examples of \( g(S, \ast) \) are shown in Fig. A.6. The transition graph for a strategy set is constructed in the following way: If an action at one of its nodes (i.e. history profiles) is determined, two outgoing links are added at the node. They are two because the co-player’s choice can be either \( c \) or \( d \), which leads to a different history profile at the next time step. If we have not determined the action, the node has no outgoing links.
The existence of negative cycles in \( g(S, \ast) \) is judged by the Floyd-Warshall (FW) algorithm [Hougardy 2010]. The FW algorithm finds the minimum distance for every pair of nodes which do not belong to a negative cycle. By distance, we mean the relative payoff difference between the players, so that outgoing links from ‘positive nodes’ \((\ast \ast d, \ast \ast c)\) and ‘negative nodes’ \((\ast \ast c, \ast \ast d)\) contribute +1 and −1 to the distance, respectively. The other nodes such as \((\ast \ast c, \ast \ast c)\) and \((\ast \ast d, \ast \ast d)\) contribute zero and will be called ‘neutral’. Specifically, we use the following algorithm:

1. Start from the root vertex of the tree. Define \( \delta \) as a 64 × 64 matrix of the minimum distances for all node pairs. All its elements are formally regarded as +∞ at the root vertex, where no links exist yet.
2. Move to one of the child vertices, whose corresponding strategy set is denoted by \( S \), by determining an action at node \( k \). This corresponds to adding two links to \( k \), and we denote these links as \( k \rightarrow u \) and \( k \rightarrow v \), respectively.
   (a) Update the minimum distance between \( k \) and an arbitrary node \( j \) by calculating \( \delta_{kj} = \min\{\delta_{kj}, \delta_{ku} + \delta_{uj}, \delta_{kv} + \delta_{vj}\} \).
   (b) For the every other pair of nodes \( i \) and \( j \), update their minimum distance by calculating \( \delta_{ij} = \min\{\delta_{ij}, \delta_{ik} + \delta_{kj}\} \).
   (c) If the updated matrix \( \delta \) has a negative diagonal element, a negative cycle exists in \( g(S, \ast) \). Do not go deeper into this branch. Otherwise, proceed to one of the grandchild vertices recursively as in the depth-first search.
3. Check the other child vertex in the same way.

To discard strategies that are not defensible as early as possible, we should begin by checking actions that are likely to form negative cycles: If a negative node exists with undetermined actions, this should be checked first, by adding outgoing links to the node. For example, in Fig. A.6, we can say that a strategy violates defensibility if it prescribes \( c \) at its unique negative node \((c, d)\) because such prescription forms a negative cycle of \((c, d) \rightarrow (c, d) \rightarrow \ldots\). Provided that \( d \) is the correct action at \((c, d)\), let us proceed to one of the subsequent nodes, \((d, d)\). One must choose \( d \) here: Otherwise, we will see a negative cycle \((d, d) \rightarrow (c, d) \rightarrow (d, d) \rightarrow \ldots\). After determining these two actions, the strategy set \( S \) can be written as \((\ast d, \ast d)\), and it is no longer possible to form a negative cycle at this point: To revisit the negative node \((c, d)\) to complete a cycle, one must go through the positive node \((d, c)\). We can say that all the possible cycles from the negative node have been neutralized.
in this strategy set \( S = (\ast d, \ast d) \). With just two steps, this procedure gives the list of memory-one strategies that satisfy denfensibility. In general, we will use the following procedure:

1. Determine actions at all the negative nodes, among which \( d \) must be chosen at \((c\cdots c, d\cdots d)\) for obvious reason.

2. If we have not determined action at a node, we will call the node ‘susceptible’. Let \( K \) be the set of susceptible nodes linked from the negative nodes.

3. For each \( k \in K \), compute \( y_k \equiv \min_{j \in G} \delta_{jk} \), where \( G \) is the set of negative nodes.
   - If \( k \) is a positive node with \( y_k = -1 \), remove it from \( K \) because this path is neutralized.
   - Otherwise, add two outgoing links to \( k \) by determining an action. Replace \( k \) in \( K \) by its subsequent susceptible nodes.

4. Repeat Step 3 until a negative cycle is found or \( K \) becomes empty. In the latter case, all the strategies in the remaining strategy set does not have a negative cycle.

**Appendix A.4. Checking the efficiency criterion for a strategy set**

Similarly, a necessary condition exists for a strategy set to satisfy the efficiency criterion. First, an efficient strategy needs to recover mutual cooperation against one-bit error at least. The transition from state 8 \((ccd,ccc)\) and state 1 \((ccc,ccd)\) must eventually reach state 0 in \( g(S,S) \). Otherwise, it cannot be efficient. This judgement is useful for a strategy set as well: We construct a graph \( g(S,S) \) for a strategy set \( S \), which is defined as the largest common subgraph of \( g(S,S) \) for every member strategy \( S \in S \) (Fig. [A.7]). For example, if we trace the transition from state 1 or 8 in this graph and find a cycle other than 0, all the strategies in \( S \) cannot be efficient, and thus it is not necessary to go further than this strategy set.

The above method checks whether the mutual cooperation is tolerant against one-bit error, which is a necessary condition for efficiency. To assure the efficiency of a strategy set, we need to take into account higher-order terms of \( e \) as well. The key observation is that only SCC’s can occupy finite stationary probability, which is an essential object in judging efficiency. By extending the graph-theoretic method in Sec. [Appendix A.1], we have
developed a method to judge efficiency for a strategy set as follows: Let \( C \) denote the set of nodes constituting the SCC’s in \( g(S, S) \). We go down the tree until every node in \( C \) has a prescribed action. Then, the SCC’s of \( g(S, S) \) will be identical to those in \( g(S, S) \) for every \( S \in S \). When this is the case, we use the following algorithm to test the efficiency of \( S \):

1. Calculate \( C \) for \( g(S, S) \).
2. Construct \( g_0 = g(S, S) \).
   - If \( 0 \to i \) for a certain node \( i \), the strategy is inefficient.
   - If \( \Omega_0^{(0)} = \{ i \in C | i \to 0 \text{ and } 0 \not\rightarrow i \} \) equals \( C \), the strategy is efficient.
   - If \( \Omega_0^{(0)} \) is a strict subset of \( C \), the efficiency criterion is undecidable from \( g_0 \). Go to the next step with \( \nu = 1 \).
3. Construct \( g_\nu \) by adding links from every sink \( \gamma \) surviving in \( g_{\nu-1} \) to the member nodes of \( \rho_\gamma^{(\nu)} \). If \( g_\nu \) has an unfixed node which is reachable from \( C \), fix the action at the nodes and then apply the same sequence recursively to its child strategy sets.
   - If \( 0 \to i \) for a node \( i \) outside \( \Omega_0^{(\nu-1)} \), the strategy is inefficient.
If $\Omega^{(\nu)}_0 \equiv \Omega^{(\nu-1)}_0 \cup \{ i \in \mathcal{C} | i \rightarrow 0 \text{ and } 0 \not\rightarrow \nu i \}$ equals $\mathcal{C}$, the strategy is efficient.

- If $\Omega^{(\nu)}_0$ is a strict subset of $\mathcal{C}$, the efficiency criterion is undecidable from $g_\nu$. Go to the next step.

4. Increase $\nu$ by one, and go back to the previous step.

This algorithm always ends with a decision between ‘efficient’ and ‘inefficient’ as in the case of the algorithm for a single strategy.

Appendix A.5. Overall workflow

Another tip to reduce the number of strategies is checking the efficiency and defensibility criteria simultaneously. While the number of strategies satisfying either one of the criteria is enormous, the number is significantly reduced by checking for the efficiency and the defensibility criteria simultaneously because these two criteria require apparently contradictory behaviours as is shown in Fig. 1. The overall workflow is thus organized as follows:

1. Traverse the strategy tree with checking the defensibility criterion. The traversal goes down to a certain depth $D_d$.
2. Traverse the strategy tree with checking the efficiency criterion. The traversal goes down to $D_e$.
3. Repeat the above two steps with changing parameters.

If $D_d$ or $D_e$ is large, the number of strategy sets increases exponentially. Switching between these two steps with small depths is important to carry out the calculation in practice. We have tested various values of $D_d$ and $D_e$, and found that it does not change the resulting number of successful strategies. We have also checked defensibility and efficiency of strategies that are randomly chosen from our calculation result. In short, we can say that the algorithm works as intended.

Appendix B. Examples

To understand the mechanisms for successfulness in detail, we will give two examples of successful strategies, denoted by ES1 and ES2, respectively. The former is taken from the strategies having the shortest recovery path and the latter is from those with a longer recovery path.

The first example of memory-three successful strategies is defined by Table B.3 and denoted as ES1. The behaviour of this strategy is distinct from
Table B.3: Action table of the first example of memory-three successful strategies, which is denoted as ES1.

| $A_{t-3}A_{t-2}A_{t-1}$ | $B_{t-3}B_{t-2}B_{t-1}$ |
|-----------------|-----------------|
| $ccc$           | $c$             |
| $ccd$           | $d$             |
| $cdc$           | $d$             |
| $cdd$           | $c$             |
| $dcc$           | $d$             |
| $dcd$           | $c$             |
| $ddc$           | $d$             |
| $ddd$           | $d$             |

TFT-ATFT in several respects: Let us look at the mechanism to stabilize the cooperation. When the players using this strategy, the mutual cooperation is recovered from a one-bit error in two steps, as shown in the first entry of Table B.3.

Although mutual cooperation is robust against a one-bit error, it does not assure that the strategy meets the efficiency criterion. This is because mutual defection is also robust against a one-bit error: When an implementation error occurs at state $63 = (ddd,ddd)$, the state eventually returns to mutual defection as

$$ (ddd, ddc) \rightarrow (ddd, dcd) \rightarrow (ddd, cdd) \rightarrow (ddd, ddd). \quad \text{(B.1)} $$

This error robustness of mutual defection is not found in TFT-ATFT. To see how the efficiency criterion is satisfied, we need to look at higher-order transitions mediated by more than one errors. Figure B.8 shows the transition between the cycles in $g(ES1, ES1)$. The graph has four cycles: (i) mutual cooperation ($ccc, ccc$), (ii) mutual defection ($ddd, ddd$), (iii) TFT retaliation ($cdc, dcd$) $\leftrightarrow$ ($dcd, cdc$), and (iv) synchronous repetition of cooperation and defection ($cdc, cdc$) $\leftrightarrow$ ($dcd, dcd$). The transition from mutual cooperation to mutual defection occurs with $O(e^3)$ while the transition for the opposite direction happens with $O(e^2)$. In other words, the net probability flow is towards mutual cooperation, whereby the efficiency criterion is fulfilled. This is distinct from the case for TFT-ATFT, which does not exhibit (iv). The transitions between these cycles for TFT-ATFT are also drawn in Fig. B.8.
Because the transition from (ii) or (iii) to (i) occurs with $O(e)$, it is sufficient to make the mutual cooperation tolerant against one-bit error to assure efficiency.

It is also instructive to convert a strategy defined by an action table to an automaton having the minimal number of states (Murase and Baek, 2019). Figure B.9 shows an automaton derived from ES1, which has 15 internal states. Compared to the automaton for TFT-ATFT (Murase and Baek, 2019), it has a greater number of states with a very different graph structure. Actually, it bears more similarity to that of a successful strategy for the three-person PG game (see the dashed boxes in Fig. B.9), and it is not a coincidence: Both of these two strategies have $m = 3$, and mutual cooperation is robust against two-bit errors, whereas the mutual defection is robust against one-bit error (Murase and Baek, 2018).

Another example strategy (ES2) is defined by Table B.4, whose automaton representation is given in Fig. B.10. Obviously, this is not a variant of the TFT-ATFT strategy, and the path to recover mutual cooperation is much
Figure B.9: Automaton representations of ES1 (left) and a successful strategy for the three-person PG game (right). Colours of nodes represent prescribed actions at the corresponding history profiles: Cooperation (defection) is prescribed at blue (orange) nodes. The label on each edge means the actions taken at the last time step, \((A_{t-1}, B_{t-1})\) (or \((A_{t-1}, B_{t-1}, C_{t-1})\) in case of the three-person game). Note the similarity between the two strategies as indicated by the dashed boxes.

longer than that of ES1 or TFT-ATFT:

\[
(1, 8) \to (10, 17) \to (14, 35) \to (16, 2) \to (3, 13) \to (14, 19) \to \\
(16, 0) \to (12, 1) \to (25, 1) \to (59, 1) \to (19, 10) \to (3, 14) \to \\
(14, 21) \to (21, 10) \to (0, 0).
\] (B.2)

Efficiency of ES2 is explained by Fig. B.8 which depicts strongly connected components in \(g(ES2, ES2)\) and transition among them. Defensibility is verified by Fig. B.10 because it has no negative cycle. The distinguishability criterion is also satisfied because of the cycle \(14 \to 16 \to 12\), with which ES2 can repeatedly exploit an AllC player.

So far, we have focused on distinguishability only against AllC players. Generalizing this idea, we can think of a strategy that can distinguish not only AllC but also a broader class of non-defensible strategies. Let us take WSLS as an example of non-defensible strategies. When TFT-ATFT meets WSLS, they do not achieve full cooperation, but they get the same long-term payoff when \(e \to 0\), indicating that TFT-ATFT cannot distinguish a WSLS player. On the other hand, ES1 is able to distinguish a WSLS player in the sense that the long-term payoff of ES1 is strictly higher than that of WSLS, and ES1 satisfies the extended distinguishability criterion. Finally, when ES2 plays against WSLS, they form full cooperation. Thus, these three successful strategies, TFT-ATFT, ES1, and ES2, show different behaviours against WSLS.
Table B.4: An action table of the second example of memory-three successful strategies (ES2).

| $A_{t-3}A_{t-2}A_{t-1}$ | $B_{t-3}B_{t-2}B_{t-1}$ |
|--------------------------|--------------------------|
| ccc                      | c  d  d  d  c  d  d  d   |
| ccd                      | c  d  d  d  d  c  c  d   |
| cdc                      | d  c  d  c  d  c  d  d   |
| cdd                      | c  d  c  d  c  c  d  d   |
| dcc                      | d  d  c  c  d  c  d  d   |
| dcd                      | d  d  d  c  c  d  c  d   |
| ddc                      | d  d  c  c  d  d  d  d   |
| ddd                      | d  d  d  c  c  c  d  d   |

Figure B.10: Automaton representation of the second example strategy (ES2). The dashed rectangle indicates the strongly connected components of the strategy.

When ES1 and ES2 play the game, their long-term payoffs are identical because of the defensibility criterion, but their cooperation probability is below 100%. This is because their recovery mechanisms from implementation error are different. Therefore, we can conclude that different types of successful strategies do not always achieve full cooperation although each of them meets the efficiency criterion. We have already seen many types of successful strategies in memory-three strategy space. To achieve full cooperation, players need not only adopt successful strategies but also select the same type of successful strategies. The problem thus boils down to the coordination game. In the memory-two strategy space, the situation is different because every successful variant of TFT-ATFT achieves full cooperation with every
other.