Rationality in Being Irrational

Table of Contents

1. Introduction

In the infinitely complex world, there are situations where, paradoxically, irrational behavior does not rule out rationality but rather suggests looking at the bigger picture. The game-theoretic assumption of rationality implies that the agents aim to maximize their utility. Sometimes it is naïvely connected to the absolute aversion of diminishing one's own payoffs, yet it is not always the case. An example can be found in the work of Król et al. [1] which studies and experimentally proves the benefits of "burning money". It points out that payoff sacrifices are important as deliberate signals of one's further intentions, and can be a source of strategic advantage. There is a play similar to "burning money" possible in the repeated stag hunt. Instead of sacrificing beforehand, a player should deviate from Pareto dominated Nash equilibrium (NE) since it is the only way to get to the Pareto dominant NE. This essay suggests a solution in the form of irrational sacrificial move and shows the conditions under which it is profitable, most crucial of which is patience of an agent. Confirmation of that is found in the study of Al-Ubaydli et al. [2] where the experiment of people playing the repeated stag hunt is described. It states that the key trait of a successful player is not risk aversion or cognition but patience. In this essay the model of infinitely repeated game is utilized since it is more general and can represent patience via discount factors. Program simulation is used to test and analyze the solution.

2. Main part

2.1. Mathematical formalization of the problem

	a	**b**
A	x,x	w,y
B	y,w	z,z

$$x > y \geq z > w$$

In a generic stag hunt game which is described above in a strategic form players often find themselves in the Pareto dominated Nash equilibrium **Bb**. Both want to get to **Aa** but the first to move will have to lose utility. The suggested solution is the irrational sacrificial action of a player 2 (choice is only a matter of preference since the game is symmetric) to change from strategy **b** to **a** that under certain condition is overall rational, i.e. leads to equal or greater aggregated utility:

$$U_i^*(s_i^*) \geq U_i(s_i)$$

$$s_i^* = (s_i^{*(1)}, s_i^{*(2)}, \dots) = (b, \dots, a, a, \dots) \quad s_i = (b, b, b, b, \dots)$$

2.2. Mathematical approaches and methods for the solution of the problem

According to Leyton-Brown et al. [3], in infinitely repeated games the payoffs of players are described by the discounted reward function:

$$U_i(s_i) = DR_i = \sum_{j=1}^{\infty} \beta^j * r_i^{(j)}, 0 \leq \beta \leq 1,$$

$$r_i^{(j)} = u_i\left(s_i^{(j)}, s_{-i}^{(j)}\right) \in \{x, y, w, z\},$$

where the discount factor *β* can represent either the fact that the agent cares more about his well-being in the short term rather than in the long term or the probability of the game being stopped in any given round *(1-β)*. In order to connect *β* with the amount of played games, the risk indicator RI_i is introduced. It represents the minimum probability player *i* still considers safe:

$$\beta^{K_i+S_i+N_i} \geq RI_i \;\rightarrow\; K_i + S_i + N_i \geq \log_\beta RI_i$$

K_i is the number of rounds player *i* keeps playing in **Bb** to demonstrate rationality (a minimum of 1 because players are “stuck” in it). If *i* changes her strategy before having established credibility, an opponent might think that she is irrational and might do it again even if she loses utility by doing so. S_i is the maximum number of rounds stalled in **Ba** player *i* accounts for (a minimum of 1 since the situation of simultaneous change of strategies is desirable yet unlikely to happen and thus is not considered). It portrays the reaction time of a player trying to reason about *i’s* motives and integrity. N_i is the number of rounds played in **Aa** (a minimum of 1, otherwise there is no benefit from playing *s**). From now on, *i* indices in *K, S, N* will be omitted to simplify the already complex equations.

DR_i^* describes the payoff of a player trying to escape from the Pareto *dominated* Nash equilibrium **Bb** to Pareto *dominant* **Aa**.

$$DR_i^* = \sum_{j=1}^{K} \beta^j * z + \sum_{j=K+1}^{K+S} \beta^j * w + \sum_{j=K+S+1}^{K+S+N} \beta^j * x + \sum_{j=K+S+N+1}^{\infty} \beta^j * x$$

DR_i is the payoff of a player who chooses to remain in **Bb**.

$$DR_i = \sum_{j=1}^{\infty} \beta^j * z = \sum_{j=1}^{K} \beta^j * z + \sum_{j=K+1}^{K+S} \beta^j * z + \sum_{j=K+S+1}^{K+S+N} \beta^j * z + \sum_{j=K+S+N+1}^{\infty} \beta^j * z$$

$$DR_i^* - DR_i = \sum_{j=K+1}^{K+S} \beta^j * (w-z) + \sum_{j=K+S+1}^{K+S+N} \beta^j * (x-z) + \sum_{j=K+S+N+1}^{\infty} \beta^j * (x-z)$$

The last member of this sum is not particularly interesting since the play should pay off earlier, because with $j > K + S + N$ the probability of the continuation of the game is lower than the risk indicator, which means planning is not considered. In order for *s** to be preferred over *s*, the condition $DR_i^* - DR_i \geq 0$ must be satisfied:

$$\sum_{j=K+1}^{K+S} \beta^j * (w-z) + \sum_{j=K+S+1}^{K+S+N} \beta^j * (x-z) \geq 0$$

$$\sum_{j=K+S+1}^{K+S+N} \beta^j * (x-z) \geq \sum_{j=K+1}^{K+S} \beta^j * (z-w)$$

$$\beta = 0 \colon \nexists K, S, N; \qquad \beta = 1 \colon\ N * (x-z) \geq S * (z-w) \ \rightarrow\ N \geq S * \frac{(z-w)}{(x-z)}$$

If the discount factor is 0, there is no sense in carrying out strategy profile s_i^*, since the game is played once. With a discount factor equal to 1, the game is literally infinite and so are the payoffs. However, if the game is finished at some point *M*, it can be seen as finitely repeated *M* times. With $\beta \in (0,1)$, the equation cannot be simplified any further. The verbal representation of conditions is straightforward: the amount of rounds played (including keeping and stalling rounds) should be at least enough to make the benefits of being in equilibrium **Aa** equal to the losses of being in **Ba**.

2.3. Analysis of the solution

There are four types of players considered:

1) the rational – play their best responses to the previous move of the opponent, $K = S = 1$;
2) the scared/risk-averse – always play maximin (in that case, **B**, earning at least *z*), $S = \infty$;
3) the unsure – someone between 1) and 2), $K, S \in (1; \infty)$;
4) the mixed Nash equilibrium players – play MNE, probabilities depend on *x*, *y*, *z*, *w*.

In order to implement the suggested strategy profile, relatively big discount factors are needed: they must provide at least three rounds for the rational and even more for the unsure. There is little need in simulating cases 1 and 2. Playing *s** against the rational will always be better than playing *s* because of the fast reaction. On the other hand, a scared partner will make playing *s** worse even if a backswing mechanism is added because playing *s* will still be better by $Sb * (z - w)$, where *Sb* is the number of rounds being stalled in **Ba** before returning to **Bb**. Case 3 simulation is needed because of the diversity of the parameters. Case 4 needs to be simulated due to the random nature of MNE. The results are presented in the table below.

Table – The results of simulation with *x=3, y=2, z=1, w=0*

	β		**Win Rate**				**Played**		
	Min	**Max**	**Min**	**Max**	**Mean**	**CV**	**All**	**WR<50%**	**Rate**
Unsure	0.817	0.990	0.454	0.994	0.815	0.1471	23366	106	0.45%
MNE	0.816	0.990	0.526	0.829	0.669	0.1040	23078	0	0.00%

The situation is winning when the accumulated utility of playing $s*$ is no less than the accumulated utility of playing s. For each β all possible K, S pairs are simulated 10,000 times to weaken the effect of pseudorandom generator flaws. Modeling is done for $\beta \in [0.001, 0.990]$ with step 0.001. $\beta \in [0.991, 0.999]$ is not considered due to the technical restrictions. The results show that $s*$ is no worse than s 74.3% of the time, provided discount factors are high and the *DR* condition is met. The code is in the Appendix.

3. Conclusion

The essay supports the views of Al-Ubaydli et al. [2] regarding patience but dissents in terms of risk-aversion insignificance utilizing the novel modeling approach. The results prove that playing $s*$ is preferred to playing s since almost always the former is either as good or better. Unexpected MNE results are tied to the weak winning condition: the difference between payoffs most of the time tends towards zero. This work may have numerous applications in real life. For example, it may help solve the dilemma of intensive/extensive growth where two energy giants choose between investing in renewable energy sources and further increasing the exploitation of non-renewable ones. If both commit to the latter, they will eventually run out of resources. Discoordination leads to one player losing the market while the other will still eventually close down due to the absence of resources. On the other hand, coordination in terms of shifting the focus from increasing the existing market share to the development of alternative fuels is the only sustainable strategy and the most preferred if the companies are concerned with their longevity. Suggestions for further research include real-life experiments to obtain the actual values of *K, S*.

4. References

1. Król M., Król M.E. On the strategic value of 'shooting yourself in the foot': an experimental study of burning money. // International Journal of Game Theory. – 2019. – DOI: 10.1007/s00182-019-00673-5.
2. Al-Ubaydli O., Jones G., Weel J. Patience, Cognitive Skill and Coordination in the Repeated Stag Hunt. // Journal of Neuroscience, Psychology, and Economics. – 2011. – DOI: 10.2139/ssrn.1764272.
3. Leyton-Brown K., Shoham Y. Essentials of Game Theory: A Concise, Multidisciplinary Introduction. – 2008. – DOI: 10.2200/S00108ED1V01Y200802AIM003.

5. Appendix

Listing 1 – Simulation of playing strategy profile *s** and comparison of utilities with *s* in Python

```
#!/usr/bin/python3

import random as r
import numpy as np

'''
This code is a simulation of infinitely repeated Stag Hunt with discount factor beta.
Strategic form of the played game (agent 1 on the left, agent 2 on the top):
             a     b
        A | x,x | w,y |
        B | y,w | z,z |
            x>y>=z>w
The game is played infinitely yet there is a possibility of it ending in any given round
equal to (1-beta).
Game is symmetrical, so agents are interchangeable (via changing the case of the strategy
names).
The aim is to illustrate the benifits of escaping equilibruim _Bb via making sacrifice
changing from _b to _a.
It is not a complete simulation featuring actual agents since there is no complex
strategic interaction needed:
    -agent 1 plays _B K+S times and then changes to _A until the end of the game;
    -agent 2 plays _b K times and then changes to _a until the end of the game.

In order to prove that this move is worthy, agent 2 must get more than the player
remaining in _Bb: during K they get z, during S - w, during N - x.
The only special case is playing with the mixed Nash equilibrium player: during K - y
with probability p or z with (1-p), during S,N - x with p or w with (1-p). Probability
p depends on payoffs as p/(1-p)=(x-y)/(z-w) or p=(z-w)/(z-w+x-y).
Player remaining in Bb gets j*z, where j is the number of the last round.
'''

(x, y, z, w) = (3, 2, 1, 0)
betas_rounds = {}


def true_with_prob(prob):
    return r.random() < prob


def test_prob(prob):
    j = 1
    while True:
        if true_with_prob(1 - prob):
            return j
        j += 1


def play_normal(beta, K, S):
    (k, s) = (K, S)
    j = 1
    sum_util = 0
    while True:
        if k > 0:
            sum_util += z
            k -= 1
        elif s > 0:
            sum_util += w
            s -= 1
        else:
            sum_util += x

        if true_with_prob(1 - beta):
            return sum_util >= j * z
        j += 1
```

```
def play_vs_MNE(beta, K, S):
    p = (z - w) / (x - y + z - w)
    (k, s) = (K, S)
    sum_util = 0
    b_sum_util = 0
    while True:
        if k > 0:
            if true_with_prob(p):
                sum_util += y
                b_sum_util += y
            else:
                sum_util += z
                b_sum_util += z
            k -= 1
        else:
            if true_with_prob(p):
                sum_util += x
                b_sum_util += y
            else:
                sum_util += w
                b_sum_util += z

        if true_with_prob(1 - beta):
            return sum_util >= b_sum_util


def rounds_from_beta(beta):
    T = 10000
    if beta not in betas_rounds:
        a = 0
        for i in range(T):
            a += test_prob(beta)
        betas_rounds[beta] = a / T
    return int(np.round(betas_rounds[beta], 0))


def sum_condition(K, S, beta):
    N = rounds_from_beta(beta) - (K + S)
    if N <= 0:
        return False

    # inclusive ranges
    left = (x - z) * sum([beta ** j for j in range(K + S + 1, K + S + N + 1)])
    right = (z - w) * sum([beta ** j for j in range(K + 1, K + S + 1)])
    return left >= right


def simulate(beta, times, play):
    total_rounds = rounds_from_beta(beta)
    K_S_winrates = {}

    # both min for unsure is 2, rational - 1
    (k, s) = (2, 2)
    for K in range(k, total_rounds - s):
        for S in range(s, total_rounds - k):
            if sum_condition(K, S, beta):
                wins = 0
                for i in range(times):
                    if play(beta, K, S):
                        wins += 1
                K_S_winrates[(K, S)] = wins / times
            else:
                break
    if not K_S_winrates:
        return None
    return K_S_winrates
```

```
def calculate_winrates_for_beta_range(start, stop, step, TIMES, play):
    beta_winrates = {}
    for beta in np.arange(start, stop, step):
        beta_winrates[beta] = simulate(beta, TIMES, play)
    return dict(filter(lambda elem: elem[1] is not None, beta_winrates.items()))


def main():
    TIMES = 10000

    beta_winrates  =  calculate_winrates_for_beta_range(0.001,  0.991,  0.001,  TIMES,
play_normal)
    print(beta_winrates)

    #  beta_winrates_unsure=calculate_winrates_for_beta_range(0.7,0.999,0.001,  TIMES,
play_normal) #in simulate k,s=2,2
    #   beta_winrates_MNE=calculate_winrates_for_beta_range(0.7,0.999,0.001,   TIMES,
play_vs_MNE)
    # beta_winrates_rational=calculate_winrates_for_beta_range(0.7,0.999,0.001, TIMES,
play_normal) #in simulate k,s=1,1

    return 0


if __name__ == '__main__':
    main()
```

Listing 2 – CSV-formatter of simulation results in Python

```
#!/usr/bin/python3

import csv
from sys import argv, exit

def print_beta_dict(betas):
    for beta in betas:
        for KS in betas[beta]:
            print("{}\t{}\t{}".format(beta, KS, betas[beta][KS]))


def csv_beta_dict(betas):
    with open(argv[2], mode='w') as f:
                w      =      csv.writer(f,      delimiter='\t',      quotechar='"',
quoting=csv.QUOTE_MINIMAL)
                w.writerow(['Beta', 'K, S', 'Winrate'])
                for beta in betas:
                    for KS in betas[beta]:
                        w.writerow((beta, KS, betas[beta][KS]))


def main():
    if len(argv) != 4:
        exit(1)

    result = {}
    with open(argv[1]) as f:
        result = eval(f.readline())

    print(len(result))
    if argv[3] == 'p':
        print_beta_dict(result)
    elif argv[3] == 'c':
        csv_beta_dict(result)

    return 0


if __name__ == '__main__':
    main()
```