**Game description for developer**

Date: 31 – Mar – 2015

Author: Dang Quang Vinh

The document tend to be updated based on the version of zTree program.

All information in the document are based on the zTree file “combine\_game\_with\_trust\_and\_contract.ztt”

**1. Introduction**

The zTree program is designed for all 4 games: simple game, game with trust score (Trust Game), game with contract (Contract Game) and game with both contract and trust score (Combine Game).

The game flow will be:

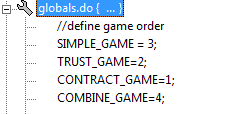
- The instructor started zTree program.

- The players use zLeaf on the client computers to play all 4 games until finish. There is no break or intervention from instructor, except for error case of program.

- All users are synchronized at the beginning of each period. It means, the users has to wait to all other users to start a new period.

**2. Game order**

The game order is defined in the program as following:

In this case, the user will play first Contract Game (number 1), then Trust Game (number 2), and so on …

The instructor can modify the order, but he must make sure that 4 games will contain distinct values from 1 to 4.

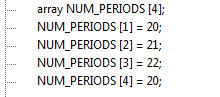
**3. Number of periods**

A period is defined as one time a user send and receive money. A game (Simple Game, Trust Game, …) contains multiple periods.

Currently, the numbers of periods are defined as: 20, 21, 22, and 20, respectively for first game, second game, and so on.

We only care about first 20 periods. The variety 21 or 22 have been used to create stochastic procedure, so users can not know how many periods they will play.

The instructor is free to modify the number of rounds, but all of them should greater or equal 20.

The number of periods are defined in the code block like below.

**4. Simulator type**

We defined three types of simulator:

**4.1. Simple Game**

- Good guy, who will send back around[[1]](#footnote-1) 80% of what he received.

- Bad guy, who send back around 20%.

- Normal guy, who send back around 50%.

**4.2. Game with contract[[2]](#footnote-2)**

If the user accepted the contrat, call trust\_partner is the trust score of the user on the partner[[3]](#footnote-3), and trust\_user is the trust score of the partner on the user.

The probability each simulator type will follow the contract is:

- Good guy:

(trust\_partner + trust\_user) / 4 + 2/3

- Normal guy:

(trust\_partner + trust\_user) / 4 + 1/2

- Bad guy:

(trust\_partner + trust\_user) / 4 + 1/3

If the simulator *does not* follow the contract, he will send 0 or 1.

Please note that, in extremce case, the probability of good guy follows the contract is more than 1 (I.e, he will always follow the contract). It is the situation when trust score of all participants (user and simulator are very high).

Also, if the trust score is higher, more likely the simulator will follow the contract.

**4.2.1. Proposing contract**

The simulator proposes the contract based on the trust score of simulator on the user.

- The sender will send round (trust\_score\_of\_simulator\_on\_user \* 10) + 1

- The receiver will send back around two times of sending amount, with variance + 1.

For example, if trust score of the simulator on the user is 0.37, the simulator will propose a contract as following:

- The sender will send 4.

- The receiver send back 8.

or

- The sender send 5 (4 + 1)

- The receiver send back 9 (2 \* 5 – 1)

or

- Sender send 3

- Receiver send 7

The probability of each values is (for the above example): 50% for 4, 25% for 3 and 25% for 5, and same for the receiver[[4]](#footnote-4).

**4.3. The probability of simulator apperance**

To prevent effect of beginning periods in a game, we set up the experiment like this for each game:

- For first 5 rounds, user always play with normal guy.

- For last 15 rounds, user have the probability 1/3 to play with each simulator guy.

**5. Trust Formula**

The system maintains two arrays for each user, one is trust score of him to other users, one is trust score of other users on him.

Initially, two arrays are defined with default value for all elements are 0.5.

For each period, let's say the user play with the partner ID 2, so we only care about the trust score of the user and the partner ID 2.

After the period finishes, we update the trust score of partner 2 on the user as following:

- If the user is sender, and the user send x to the partner:

trust\_current\_period = x / 10

(with 10 is the maximum the sender can send).

- If the user is receiver, suppose he receive first a from the partner, then send back b.

We define the medium value for b, which b = 2 \* a, means “fair” response, because if the receiver send back the double of money he received, the result is he both players got the same positive amount.

So, if the receiver send back b = 2 \* a, he got trust score of this round is 0.5

If he send back 0, obviously he got trust score 0.

If he send back maximum he can (3 \* a), he got trust score 1.

So, the formula to calculate trust score is:

If b < 2 \* a:

trust\_current\_period = b/(4\*a)

If b >= 2 \* a:

trust\_current\_period = b/(2\*a) – 1/2

After calculate the trust score for the last period, we update the total trust score by aggreating function:

total\_trust = total\_trust \* 0.5 + trust\_current\_period \* 0.5

Example**:**

Initially trust score of the user from partner 2 is 0.5

The first round: the user is sender, and he send 8.

trust\_current\_period = 8 / 10 = 0.8

total\_trust = 0.5 \* 0.5 + 0.8 \*0.5 = 0.65

Second round: the user is receiver (still play with partner ID 2), he receive first 6, then send back 10. Because 10 < 2 \* 6 so:

trust\_current\_period = (¼) \* (10 / 12) = 5 / 24 ~ 0.208

total\_trust = 0.65 \* 0.5 + 0.208 \* 0.5 = 0.429

In game with contract:

If the player follows the contract:

trust\_current\_period = 1

otherwise, his trust score from all partner will reduce 10%.

Example:

Consider the user has three partner 1, 2 and 3, with trust score from them on him are 1, 0.5 and 0.8 respectively.

If the user plays with partner 2 and he does not follow the contract, the trust score from them on him will be: 0.9, 0.45 and 0.72.

**6. Hypothesis**

Game 1:

Game 1 is the original trust game which has been studied many times. The hypothesis for this game is (0, 0), means the sender send 0 and the receiver send back 0.

Game 2:

Our hypothesis for game 2 is (5, 10).

Let's consider the sender:

- If he send more than 5, his payoff will reduce.

- If he send less than 5, his trust will reduce.

So, the Pareto maximization for him is sending 5.

Similar for the receiver, we can find the Pareto maximization for the receiver is sending back 10.

Game 3:

- The user always has to right to reject the contract, so if he accept the contract, our hypothesis predicts that he will follow the contract.

- If the user reject the contract, the game comes back to Game 1.

Game 4:

- If the user accept the contract, it should be similar with game 3.

- The difference is, how often, or with whom the user will accept the contract. Our hypothesis is: with a threshold of trust score, the user will accept the contract with the partner.

1. We use the term “around” because there is a variance in amount the simulator send. The range is + 1. For instance, if the sender send $10, the normal guy will send back $20 + 1. In the case with contract, the range is + 2. [↑](#footnote-ref-1)
2. Here, we mean game with contract in the case the user accepts the contract. If the user reject the contract, the game comes back to simple game. [↑](#footnote-ref-2)
3. I.e, how much the user trust the partner. [↑](#footnote-ref-3)
4. First, the simulator calculate: x = round (trust\_score\_of\_simulator\_on\_user \* 10), then with 50% of chance the simulator will use this value for the contract, and 25% the simulator will use the value (x – 1) for the contract, and 25% simulator of (x + 1) for the contract. So, we have the contract value for the sender, let's call y. The the simulator double the value y, then apply the same probability again for the receiver. So, there is 50% chance of using 2 \* y, then 25% of using (2 \* y – 1), and so on. [↑](#footnote-ref-4)