# Comp3021 PA3 BONUS PART

As mentioned in the Game Process Introduction, the actions of the computer players are given randomly. To improve their performance, you can design some strategies for the computer players. Please attach a report in your submission to brief introduce your implement of the strategies for computer players, please be precise in the report. We will not be able to consider your bonus work if you do not attach a report.

1.

Improve performance by implementing more complicated condition branches. When implementing processPlayerTurns(), I just code some simple cases to check validation of value and rely on the mechanism of random value generated by Java to determine the action computerPlayer takes, it is easy to implement but performs silly and randomly.

To solve this, we can implement more condition branch (if…else) to let computerPlayer act less randomly and efficiently.

There are a lot of condition branches we may implement further, here comes some simple example. (written in pseudocode)

```
While(in_computerPlayer_turn){
    If(selectedCity.armySize>neibourCity. armySize
        and neibourCity notIn cityList_of_ computerPlayer){
            chooseMostPowerfuleGeneral attack neibourCity with
            0.8*neibourCity. armySize +
            random.nextInt(selectedCity.armySize-neibourCity.
            armySize)
    }
    If(upgradeTown fails){collectTask}
    If(recruitArmy fails){improveCrop or collectTax}
    If(selectedCity.armySize>neibourCity. armySize*2
        and neibourCity In cityList_of_ computerPlayer){
```

```
                        transfer random.nextInt(selectedCity.armySize) from
                        selectedCity to neibourCity
                }
                ……
        }
```

We can implement more condition check like these, letting computerPlayer take action a little smartly basing on current information and condition of cities computerPlayer owns.

2.

Improve performance by imitation, basing on strategies 1, implement a method which generates new file to record a series of actions acted by humanPlayer and computerPlayer and towns' current information. And computer may load these file and then randomly choose one file follow in the upcoming new game.
Further explanation:

if human player won the game, we mark file recording human player's action and town's condition step by step as HW_index.txt (meaning human_win_0) and mark other file recording computer player's action and town's condition step by step as CL_index.txt (meaning computer_lose_0).

if computer player won the game, we mark file recording computer player's action and town's condition step by step as CW_index.txt (meaning computer_win_0) and mark other file recording human player's action and town's condition step by step as HL_index.txt (meaning human_lose_0).

2.1. Learning from computer own experience, when new games reboot, computer can load one of CW_index.txt and follow action recorded in file that leading to victory in the past.

2.2. Learning from human players, when new games reboot, computer can load one of HW_index.txt and following action recorded in file, challenging human player with his/her own previous strategies. (It is similar to old Chinese saying, to treat people with their own ways.)

To improve interaction of game and performance of computerPlayer, exchanging the player of first turn every new game is worthwhile to implement.

3.

Reinforce learning(refer to https://course.cse.ust.hk/comp4211/rl-introx.pdf), we can set up model of reinforce learning if human player have played this game many times because of many data file having generated (training set).
In these game, we have following elements of reinforce learning:
Goal: getting all town as fat as possible
State: the current information of towns
Actions: collect tax, upgrade town, improve crop field, recruit army, send troop, do nothing.
Rewards: high positive value in the short path to victory, low positive value in the long path to victory, negative value in path to failure.

Read all files (CW_index.txt, HW_index.txt, CL_index.txt and HL_index.txt), constructing reinforce learning framework, to get highest reward in processing training data(game record file), computer may tend to take actions in the short path to victory. Eventually, after learning from all training data, computer would get a reinforce policy(e.g. strategies to play game) and computer player can take action predicted by policy.