Summary of previous work

Lukas Pestalozzi

February 2017

1 Tichu the game

1.1 Tichu and AI

- game of not complete information
- 4 players game (2 against 2 ,there is a partner and 2 enemies)
- Following situations demand different decisions to be taken:

  Standard situation: play or pass on a already played trick.

  First play: play on an empty table (first trick of a round)

  Card swap: At the beginning of each game, decide which card to give to each player.

  Tichu announcement: Before playing the first time in a game, decide wheter to announce a Tichu.

  Grand Tichu announcement: Before receiving the 8th card, decide wheter to announce a grand tichu.

  Whishing a card: After playing the Mah Jong, the player may wish a card to be played.

  Dragon trick: When taking a trick with the dragon, decide whom (of the enemy players) to give that trick.

2 Papers concerning Tichu or similar games

2.1 Eine künstliche Intelligenz für das Kartenspiel Tichu (2013)

This is a thesis from the University of Stuttgard (2013) by David Pfander. David created an AI that is able to play and compete with human players. His approach is to hand-craft a *GetUtility* function that gives a value to a partition of a set of cards. The higher the value, the better the partition of cards. This *GetUtility* function is rather involved and relies on several hand-tuned threshholds and parameters.

Description

Due to the partial unobservability of the game (and the large number of possible card distributions of the other players), David decides to use a minimax-search of depth 1. Ie, taking the action directly maximising the *GetUtility* function without calculating possible moves for the other players.

An action is defined in terms of *partitions*. A partition follow the standard mathematical definition of a partition (partitioning a set S in disjoint sub-sets (may include the empty-set) of S, such that the union of all elements in the partition forms S again).

Def1 (valid partition): A partition of tichu-cards is valid iff all elements of the partition form a valid combination of cards according to the tichu rules.

Def2 (valid action): An *action* is the tuple (p, c) where p denotes a valid partition of tichu cards, and $c \in p$ such that c is playable in the current game state according to the tichu rules. Note that $c = \emptyset$ denotes the *pass* action.

$\mathit{GetUtility} : (p, c) \rightarrow R$ is the function giving each action a numeric value depending on the current game state.

With this function, the different decisions can be taken as follows:

Standard situation: Find the tuple (p, c) of the players hand-cards that maximises *GetUtility*.

First play: Find the tuple (p, c) of the players hand-cards that maximises *GetUtility* $(c \neq \emptyset)$

Card swap: For all possible 3-tuples of the players hand-cards, find the maximum *GetUtility* for the remaining 11 cards. Then decide which of the 3 cards to give to which player. This also includes some heuristics to help the teammate and sabotage the enemy.

Tichu announcement: Find a partition with *GetUtility* higher than some predefined threshhold. Also don't say tichu when teammate already announced one, and only seldom if an enemy announced one.

Grand Tichu announcement: Same as simple Tichu, but with another threshhold taking into account that not all hand-cards are known yet.

Whishing a card: Many different strategies possible. All with the goal to disrupt the enemies chances of having a bomb.

Dragon trick: Many different possible strategies possible. David decided to give the dragon trick to the enemy player with more hand-cards, with the idea that the more cards one has, the longer it takes to finish.

The *GetUtility* function is rather sofisticated and relies on many hand-defined parameters/threshholds. I won't describe it here. David describes the algorithms he used quite well, maybe I can rebuild (or ask him for the code) to have an agent to compare.

Results

12 Persons played with and against the AI and answered some questions afterwards:

AI strengt: 8/12 rated the AI strengts as *strong* or *verystrong*. Hinting that it can compete with human players.

Comparison to other human players: 8 deemed the AI as *similar* or *stronger*. Only 3 answered with *weaker*.

how oftendid the AI play nonsensical moves: Only 2 answered *often*. The others said *never* or *almostnever*.

Conclusion

David creates the AI with a fine tuned *GetUtility* function and acheives a playstrengh similar to human players. He does a quite complete analysis of all different decisions that have to be taken during the game and incorporates the results in the *GetUtility* function.

For me, this work is very useful to understand the different nuances of the game and may yield valuable insights during the development of my AI.

2.2 Fight the Landlord (Dou Di Zhu)

This is a report by 4 (students?) about creating an AI for the game Fight the Landlord (Dou Di Zhu). There are some crucial differences to Tichu, but the overall game is the same. The most important differences are:

- Tichu is played 2 vs 2 while Dou Di Zhu is 2 vs 1.
- In Dou Di Zhu there is a bidding phase at the beginning to determine who is the Landlord (the 1 player playing alone).
- In Tichu at the beginning each player has to give one card to each other player.
- In Dou Di Zhu there are more possible card combinations that can be played (Tichu can play a subset of them).
- Tichu has 56 cards (52 normal + 4 special) while Dou Di Zhu has 52 normal + 2 jokers.
- In Tichu you can announce a Tichu.

All in all, Tichu has a smaller fanout (less possible moves) and there are more players, which means less cards per player. But the 4 special cards, the card swaping and the 'Tichu' announcement increases the tactical possibilities.

Description

Similarly to David, they split the cards into valid partitions. They define the *optimal partition* as the partition with lowest number of elements. They created 4 different Agents:

Random Agent: Plays random, valid moves.

Simple Agent: Plans for the future, plays the low-ranking valid card(s) first if possible.

Advanced Agent: Analyzes different contexts, and plays accordingly. Makes use of a decision tree and probabilities.

Predictive Agent: Remembers played cards and predicts the remaining cards of the other players. Plays cards as optimal as possible.

2.2.1 Results

Their two predictive agents managed to beat a human landlord 54% of all games (the advanced agents managed to win 52% of the games).

2.2.2 Conclusion

They also used smart "hand crafted" rules (a decision tree) and some probability to create their agents.

Their description of the implementation is rather shallow so it is not as useful as Davids report, but it shows again that it is possible to create an Agent capable of competing with humans.