**Cake Poker:**

**The Adjusted Winner Applied to Goods with Combinatorial Value**

**SUMMARY:**
In real-life division problems, there often is a need to divide goods which, for a particular person, may carry values which are not independent of each other. For example, in a divorce situation, a husband may think that the HDTV and the DVR are of greatest value when put together to watch football, while her wife has independent appraisals of the devices she uses for watching movies and taping soap operas. Current modeling of fair division problems does not account for this phenomenon.

For this project we created a game that tasks players with dividing a set of playing cards with the intention of forming a poker hand, in conjunction with two cards which only they know about. We decided to make this game because it creates a situation where the goods are discrete, subjectively valued, and non-independently valued. The game is also immediately understandable to anyone with experience playing card games, and involves very little abstraction on our part. With this game, and with a simple-but-effective AI, we simulated thousands of Adjusted Winner divisions and evaluated whether a fair division reliably occurred.

**LONG DESCRIPTION:**
The game we designed is called Horrendous Hog. The way it works is as follows:

- Two players each gets two Pocket cards, and eight "Cake Cards" are placed in front of them. A neutral Dealer puts seven cards face-down in front of him.
- Using a Fair Division Algorithm (we are only using Adjusted Winner), the Cake Cards are between the players. The cards each player gets is called their "Slice cards." One player may end up with more Slice cards than the other.
- For each player i, record these 3 probabilities:
  - P_i: i's chance of beating the dealer with just his Pocket cards
  - S_i: i's chance of beating the dealer with his Pocket and Slice cards
  - C_i: i's chance of beating the dealer with his Pocket and ALL the Cake cards.
- Player i's score for the game is 100 * (S_i - P_i) / (C_i - P_i).

One aspect of the game that is unintuitive is the scoring metric. Essentially, it measures where the value of your hand falls in relation to your best and worst possible hands. A score of 100 means that you would not have done better if you had any more cards from the Cake. A score of 0 means that you did as well as if you had received no cards from the cake at all. A score of fifty means that the value of your hand is halfway between no cake and all cake.
The score thus provides an analogous metric for a player’s valuation of their slice of cake. Along with it we have some analogous ways of evaluating fairness.

- Envy-freeness: If player 1’s score would be improved by trading slice cards with player 2, or vice versa, the division is not envy-free.
- Proportionality: Both players have scores above fifty.
- Equitable: Both players have equal scores.

Now, because the division involves only 8 goods which cannot be divided, so it may be impossible to satisfy these properties. But what we can do is compare the results of the Adjusted Winner algorithm to one where every one of the $2^8$ possible divisions of the cards is examined.

MODELING AND CODING
The project was coded in C#. Most of the evaluation of hands and cards was done using Keith Rule’s fantastic poker library:

codeproject.com/Articles/12279/Fast-Texas-Holdem-Hand-Evaluation-and-Analysis

For which I wrote a number of adapters. This library uses 64 bit masks to express sets of cards, making it very fast at enumeration, randomization, set operations, and more. It uses precompiled tables to identify the hand types (pair, straight, etc), and can compare one hand to another to see which one is better. It was very easy to model a deck of non-repeating cards, and to very quickly estimate $P_i$, $S_i$, and $C_i$ with a Monte Carlo simulation.

The Adjusted Winner procedure is implemented as a generic method that actually can work with any type of good or player. It follows the official procedure verbatim, with one exception: it uses coin flips to resolve situations where the players give objects equal bids.

The CakePlayer object is an object designed to be passed as a parameter to the Adjusted Winner procedure. On its own, it is unable to play Horrendous Hog, it needs to have a bid-assigning function stored in its “Strategy” field. This is a functional approach to design, but you can also think of it as being like an abstract class.

We wrote three strategies for the CakePlayer to use:

- “Human,” in which the bids are made by a human.
- “Random,” in which the bids are made randomly.
- “Polite” in which the computer picks the best hand that deviates some amount from which a typical player would pick.

The Polite strategy is computationally expensive, but it gets somewhere close to the way we expect people to actually behave in a fair division situation. The strategy takes in its own pocket, the set of Cake Cards, and also a “politeness variable” $H$ that will come up soon.
Here’s what the Polite strategy does. We’ve written it as the actions of a player named Harry, whose opponent is a player named Sally:

- For each possible slice, Harry measures the kind of hand it would create when combined with his pocket cards, and puts all 256 of these slices in order.
- Harry then removes any slices which give the same hand but require him to take more cars than he needs to. Call this resulting list $L_h$.
- Harry then makes a new list, where he sorts the remaining slices by how much he thinks Sally would value them. Because he doesn’t know what Sally’s pocket is, he estimates it via Monte-Carlo. Call this resulting list $L_s$.
- He then picks a slice that is at least $h$ indices further up $L_h$ than $L_s$.
  - Say $L_h = s_1, s_2, s_3, s_4, s_5$; $L_s = s_1, s_3, s_4, s_5, s_6, s_2$; and $h=3$. Harry chooses $s_2$ as his slice because it is 2nd in his list, but 6th in a typical player’s list.
- Harry splits his 100 points equally among every card in the slice.

Therefore, when $H=0$, the AI picks the best hand possible, but at higher values of $H$, the AI will defer its top picks if it believes that its opponent will make the same choice. Regardless of the value of $H$, this strategy is effective enough to beat both the random player and human players most of the time.

SIMULATION
We simulated 100 games each between two Polite AIs, named Harry and Sally, with the following pairs of $H$ values.

- $H = 0$ and $H = 0$
- $H = 6$ and $H = 6$
- $H = 12$ and $H = 12$
- $H = 0$ and $H = 6$
- $H = 6$ and $H = 12$
- $H = 0$ and $H = 12$

We plotted the resulting scores in scatter plots, and also measure the average score of each player over the games of each set. You can see our graphs on the next few pages. In the scatter plot, the horizontal position is Harry’s score, and the vertical position is Sally’s score. If the dots congregate over to the bottom-right corner, it means the games were mostly in Harry’s favor. If they congregate in the top-left corner, it means the games were mostly in Sally’s favor. Dots along the line $x=y$ indicate equitability, and dots in the top-right quadrant indicate proportionality.

To the right of each plot is a column graph. The blue column is Harry’s average score, and the red column is Sally’s.
These results carry an interesting message. First, it can be seen that when both players play with high levels of Politeness, the divisions tend to be more equitable, and both players end up with a higher average score. You can see this by looking at the progression from Charts 0,0 to 12,12. At first the scores veer either in one direction or the other, but when both players are willing to decline valuable grabs that aren’t valuable to just them, the results are closer to fair.

But on the other hand there’s a devious twist. When we give Harry lower values of H than Sally, the results begin to consistently come out in his favor. There are just a handful of games in Chart 0,12 where Sally gets a higher score than Harry, and it is also rare for Sally to get a score above fifty.

CONCLUSION
It can be seen from our data that when dividing goods with combinatorial values using Adjusted Winner, more fair results come when players pass up portions of the goods that they value in favor of portions that they value more than the norm. However, this strategy only works when both players adhere to it -- otherwise, the player who is least Polite gets the better end of the deal.

Crafty players will recognize that there is no advantage to being polite, and so we are back at square one (or Chart 0,0). When neither player is Polite, the best combinations are taken apart by mutual competition, and both the fairness and the average value gained by each player suffers.

One of the primary advantages of the Adjusted Winner procedure is that it rewards players who are honest about their preferences. But the strategies rewarded by the AW procedure, as well as its optimality, appear to fall apart once the goods have values that don’t combine linearly. An Adjusted Winner procedure more tailored to the real world would also reward
players who account for the differences between their preferences and the expected preferences of other players.