**HEURISTICS**

Here are the used approaches for each crucial parts of agent:

1. Uncertain Preference Profile: Copeland comparison with similarity indexing for finding weights of issue values, calculation of sum of the squared errors in frequencies for finding issue utilities, AHP
2. Opponent Modeling: Frequency Model (Updated for finding issue utilities)
3. Bidding Strategy: Trade Off Strategy, CP nets (considers also opponent modeling)
4. Acceptance Strategy: AC Next

**UNCERTAIN PREFERENCE PROFILE**

So, we are given K1, K2, K3, …, K10 and ordering is ascending. Starting from K10, all bids that have smaller utility than K10 are visited and rankings for K10 becomes K10 > K9, K10 > K8, …, K10 > K1. Continuing from K9, all bids that that smaller utility than K9 are visited and rankings for K9 becomes K9 > K8, K9 > K7, K9 > K1. And, this process continues with K8, K7, K6 etc.

To find weights of issue values, heuristic here is that since all bids are given in a sorted order, then issue values can be treated as the same way. For each issue, two-dimensional matrix that represents being big and small in terms of value comparison with the size (value size \* value) is created. The aim is to fill these Copeland matrices depending on the ordering of issue values and similarities of bids which is expressed below.

If utility of K10 is bigger than utility of K9, then the utility of values of K10 is bigger than the utility of values of K9 with the Copeland pairwise comparison. However, this pairwise comparison also checks similarities in bids. If K10 has better utility than K9, then two-dimensional issue matrix is updated with the similarity. If similarities are big between 2 bids, then we can say that different issue values have significant impact on ordering and matrix is updated with the big similarity count (issue “food” matrix [Finger-Food][Handmade Food] = this big similarity).If similarities are small between them, then the matrix is updated with this small similarity. (issue “food” matrix [Finger-Food] [Handmade Food] = this small similarity)

ISSUE 1 VALUES MATRIX (FOOD)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Chips and Nuts | Finger-Food | Handmade Food | Catering | Sum of rows |
| Chips and Nuts | 5 | 5 | 5 | 5 | 20 |
| Finger-Food | 10 | 10 | 5 | 5 | 30 |
| Handmade Food | 5 | 10 | 10 | 10 | 35 |
| Catering | 5 | 3 | 2 | 5 | 15 |
| Sum of columns | 25 | 28 | 22 | 25 |  |

ISSUE 5 MATRIX (MUSIC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MP3 | DJ | Band | Sum of rows |
| MP3 | 0 | 10 | 25 | 35 |
| DJ | 15 | 0 | 15 | 30 |
| Band | 15 | 20 | 0 | 35 |
| Sum of columns | 25 | 35 | 20 |  |

After filling the matrices, all matrices have the information about the Copeland comparison of different values of the same issue. Sum of any issue value row represents how many times the issue value is bigger than other issue values and sum of any issue value column represents how many times the issue value is smaller than other issue values.

For the issue 5 matrix, sum of rows is 35, 30 and 35. The agent divides sum of row to sum of row + sum of column to decide on “being big percentage” to find the weight of issue value. MP3’s percentage is 35/60, DJ’s is 30/65 and Band’s is 35/55. Then normalizing these values give the percentage of utility of each value. The utility of;

* 1. MP3 is (35/60)/ (35/60 + 30/65 + 35/55)
  2. DJ is (30/65)/ (35/60 + 30/65 + 35/55)
  3. Band is (35/55)/ (35/60 + 30/65 + 35/55)

This is how the agent calculates the weights of issue values.

Since the “being big” frequencies exist in Copeland matrices, agent checks how well the frequencies are distributed in each issue matrix. If the distribution of issue values is well, then we can say that the issue is not very important. If the distribution of “being big” frequencies are not good, then we can say that the issue is important. In other words, agent looks at the how well the frequencies are distributed in matrices. If the frequencies are close to matrix average for issue value, then agent says that this issue may not be very important. However, if the frequencies are far from each other, the issue can be important because some of the issue values are too many times bigger than other issue values.

Finally, the agent has weights of issues and weights of each values. So, calculating utility of any bid is done using AHP. How every step of uncertain preference profile handling is showed in Java code in Appendices with the comment "uncertain preferences”

**OPPONENT MODELING**

The main idea behind the opponent modeling is frequency model. However, there is a problem in original frequency model in calculating issue weights.[1] So, to calculate the weights of issues, different mechanism is used rather than normal bid comparison of adjacent bids. Since the frequencies are known by frequency model, agent checks how well the frequencies are distributed in each issue. If the distribution is well, then we can say that the issue is not very important. If the distribution of frequency values is not good, then we can say that the issue is important to opponent. Since some of issue values are sent again and again. In other words, agent looks at the how well the frequencies are distributed in matrices. If the frequencies are close to matrix average for issue value, then agent says that this issue may not be very important. However, if the frequencies are far from each other, the issue can be important.

The average of each issue array is taken, and sum of squared errors is calculated for each issue value in their own issue matrix with the average and saved into one dimensional array. So, in one dimensional array, we have the total sum of squared errors for each issue. If, the sum of squared errors is high, then we can say that this issue values are not distributed well, and this issue is important. Otherwise, the issue values are well-distributed and this is not is not that important. After normalization of sum of squared errors, agent gets issue weights. By the way, frequency model is updated after each received bid from the opponent. How opponent modeling is handled in Java code can be found in Appendices with the comment “opponent modeling”

**BIDDING STRATEGY**

In bidding, trade-off strategy is used. The opponent can give up easily on some issues compared to other issues. So, when opponent sends a bid, the most important issue and its value are fixed, but all other issue values are increased in terms of agent's preference profile by CP nets

Bidding strategy mainly focus on a Cp nets. In this part, Genius has really useful methods like “countEqualValues”. First, the agent compares all possible bids in the outcome space with the opponent bid. If any bid in the outcome space has the same issue value with opponent’s most important issue and its value, then this bid can be used for the future.

Idea behind all of these is that giving up on the most important issue can be hard for opponent. However, given on other issues can be easier. So, we are creating bids with the most important issue value. Then inside of these possible bids, the best bid according for agent is sent to opponent.

Opponent bid: <Cheese, mix own drinks, A tent, Design the card, *hire a DJ*, Clean Master>

Most important issue is music. Here is one possible bid that will be created by agent.

<Buy bags of chips, Soda’s and water, Dorm room, Plain, *hire a DJ*, Soap and water>

As a result, bidding strategy considers both of the opponent preference profile and agent’s preference profile.

**ACCEPTANCE STRATEGY**

The acceptance strategy is based on AC Next. If the utility of last received bid is bigger than agent’s next bid, then the bid is acceptable. However, I have added 2 more conditions here. Checking whether the opponent is coming from the better bid than the last one or not (is he trying to maximize my utility during the negotiation (is he trying to maximize my utility during the negotiation to agree upon?) and whether opponent’s new offer has lower utility than the last one for him/her. (is he/she trying to lower his/her utility during the negotiation to agree upon?)

**REFERENCES**

[1] Okan Tunali, Reyhan Aydoğan and Victor Sanchez, “[Rethinking Frequency Opponent Modeling in Automated Negotiation](https://link.springer.com/chapter/10.1007/978-3-319-69131-2_16)“, In the Proceedings of PRIMA 2017: Principles and Practice of Multi-Agent Systems, Lecture Notes in Computer Science, vol 10621. Springer, pp. 263-279, 2017.