Das Ziel ist es für Agenten A und B die minimierung der Rüst- beziehungsweise Fertigungskosten zu erreichen. Dabei müssen sich beide Agenten für eine Reihenfolge einigen. Diese Reihenfolge bestimmt die Kosten.

Der derzeitige Mediator gibt uns dabei immer die Reihenfolge {1,2,3,4,5…200} wobei genau eine Zahl mit einer anderen vertauscht wird. -> {2,1,3,4,5,6….200}(Index 0 wurde mit Index 1 vertauscht)

In der Vorlesung von Herr Prof. Homberger zum Thema Just-in-Sequence sieht man das mit Sequencen mit der Menge M = {1,2,3,4} gearbeitet wird. Die Anzahl möglicher Anordnungen von unterscheidbaren Objekten in einer bestimmten Reihenfolge für die Größe der Menge 4 ist 24. z.b {3,1,2,4}. Mit so einer geringen Menge ist die beste Lösung für jeden Agent leicht zu berechnen.

Bei unserem Agentenbeispiel erhielten wir jedoch eine Menge mit der Größe 200 M = {1,2,3,4…., 200}. Da es uns nicht möglich ist alle 200 Permutationsreihenfolge durchzugehen und somit die beste Lösung zu bestimmen war eine unserer ersten Ideen das der Mediator genau eine zufällige Permutation der Reihenfolge uns zurückgibt. Das Ergebnis war jedoch unzufrieden.

Deswegen entwickelten wir eine Heuristik die wahrscheinlich nicht die beste aber eine definitiv kleinere Lösung hergibt.

1. Fange bei costMatrix[0] an suche den kleinsten noch nicht besuchten Wert in diesem Array.
2. Bsp-> Index 47 war der kleinste noch nicht besuchte Wert -> costMatrix[0][47]
3. Setze Index 47 als besucht -> besucht[47] = true
4. Speichere Index 47 als nächster Wert für die sequence -> sequence.add(47)
5. Mache weiter mit costMatrix[47] suche den kleinsten noch nicht besuchten Wert in diesem Array.
6. Wiederhole Punkt 2-5 bis alle Werte besucht worden sind.

```
ArrayList<ArrayList<Integer>> all_sequences = new ArrayList<>();
        for(b = 0 ; b< getContractSize();b++){
            boolean[] visited = new boolean[getContractSize()];
            ArrayList<Integer> result = new ArrayList<Integer>();
            int i = b;
            int j = 0;
            result.add(i);
            visited[i]=true;
            while(!EverythingVisited(visited)){

                j = getIndexOfSmallestValueNotVisited(costMatrix[i],visited);
                i = j;
                result.add(i);
                visited[i] = true;
            }
            all_sequences.add(result);
        }
```

7. Hole aus allen gefunden Sequencen die minimalste.

Jeder Agent besitzt nun eine Sequence die er als minimal ansieht beim verhandeln ist es die einzige Sequence die beide Agenten verwenden dürfen. Proposal A sei dabei die Wunsch Sequence von AgentA. Während wir Proposal B als die wunsch sequence von AgentB ansehen.

Agent A erreicht mit ProposalB einen Wert von 10564.

Agent A erreicht mit ProposalA einen Wert von 461.

Agent B erreicht mit ProposalA einen Wert von 9520.

Agent B erreicht mit ProposalB einen Wert von 6991.

Da eine Einigung zwischen beiden Parteien erreicht werden muss. Muss einer der beiden Agenten Zugeständnisse machen während der andere einen möglichen Rabatt kriegt. Dabei muss es sich jedoch für beide Parteien lohnen.

Jeder Agent will nun seine Sequence und bietet Rabatte an.

Für AgentA ist die maximale Anzahl an Zugeständnisse 10564 – 461 = 10103

Für AgentB ist die maximale Anzahl an Zugeständnisse 9520– 6991= 2529

Beide Agenten bieten abwechselnd ihre Proposals mit den Rabatten an.

Für daten3ASupplier_200 & daten4BCustomer_200_5 erhalten wir somit die Werte

Agent A : Contract: 461 + 2530 Zugeständnis = 2991

Agent B : Contract: 9520 - 2530 Rabatt = 6990

Insgesamt also 9981 und das Proposal von Agent A wurde mit Zugeständnissen angenommen.

Da Agent A nichts von dem Evaluation Wert von AgentB mitkriegt weiß er noch nicht ab welchem Zugeständniswert AgentB annehmen wird. AgentB könnte probieren noch mehr zu pokern und nicht direkt das Angebot annehmen sobald es besser als sein eigenes Proposal ist. Würde der Wert jedoch über die maximale anzahl an Zugeständnissen kommen die sich AgentB erlauben kann so würde er selber riskieren einen verlust zu bekommen.