Paarbildung

Für die Paarbildung ist `createPairs` unsere Hauptmethode und dafür verwenden wir einen Greedy-Algorithmus. Wir lassen den User (Administrator) eine Priorität von 1(am wichtigsten) bis 3 eingeben, um die Präferenzen von Essensvorlieben, Altersdifferenz und Geschlechterdiversität zuzuordnen. Diese Eingabe lässt die Kriterien 6.6 bis 6.8 und die funktionale Anforderung 8.1.1 optimal erfüllen. Kriterien 6.9 ist erst für die Gruppenbildung relevant und Kriterien 6.10 ist eigentlich in unserem Algorithmus immer optimal erfüllt.

In der Methode `createPairs` werden zuerst alle Teilnehmer extrahiert, die noch zu keinem Paar gehört. Dann werden diese Teilnehmer in 3 Gruppen klassifiziert: WithYesKitchen, WithMaybeKitchen und WithNoKitchen. Da eine Küche für ein Paar notwendig ist und da wir die Küchen gemäß Kriterien 6.3 nicht verschwenden sollten, bilden wir die Paare in folgenden Reihenfolgen:

1. WithYesKitchen & WithNoKitchen
2. WithYesKitchen & WithMaybeKitchen
3. WithYesKitchen & WithYesKitchen
4. WithMaybeKitchen & WithNoKitchen
5. WithMaybeKitchen & WithMaybeKitchen

Diese Reihenfolge stellt sicher, dass alle Yes-Küchen vor der Maybe-Küchen benutzt werden sind. Damit die wichtigste Präferenz möglichst gut erfüllt ist, lassen wir die obige Kombinationen erstmal mit `restrictedToPriorityOnePreference` als `true` durchführen, sodass ein Paar wird nur gebildet, wenn die Priority-1-Präferenz am besten erfüllt ist (d.h. dieselbe Essensvorlieben / dieselbe Altersgruppe / verschiedenes Geschlecht). Dann werden diese Kombinationen ohne solche Beschränkung (`restrictedToPriorityOnePreference` als `false`) nochmal durchgeführt.

Danach werden die beide Listen in `handlePairMatchingWithPreference` bearbeitet. Nur wenn `restrictedToPriorityOnePreference true` ist, werden die Gruppen weiter klassifiziert und zugeordnet (z.B. beide `groups1.get(0)` und `groups2.get(0)` beinhalten nur Teilnehmer mit `FoodPreference NONE`, und beide `groups1.get(1)` und `groups2.get(1)` beinhalten nur Teilnehmer mit `FoodPreference MEAT`, usw.) sonst gibt es immer noch nur die beiden eingegebenen Listen. Zunächst werden die verteilte Gruppen von `groups1` iteriert, in dem die Teilnehmer davon auch iteriert wird. Zu jedem Teilnehmer wird eine Liste von gültigen Kandidaten gemäß Kriterien 6.1 & 6.5 erstellt, dann wird diese Liste abhängig von der eingegebenen Präferenzen anpassend sortiert. Am Ende ist der erste Kandidat der Liste die “beste” Kandidat entschieden und ein Paar ist gebildet. Nach allen Iterationen sind dann alle Paare möglichst optimal gebildet werden.

Zusammenfassung der Ergebnissen von Paarbildung:

Hier zeigen die Pärchenkennzahlen unter verschiedenen Prioritäten.

Priority 1
Priority 2
Priority 3

food1,age2,gender3:	food1,age3,gender2:
Number of Pairs: 152 Number of Successors(Nachrückende): 6 Average Gender Diversity: 0.625 Average Age Difference: 0.3223684210526316 Average Food Preference deviation: 0.0 ---Only those created by algorithm--- Number of Pairs created by algorithm: 79 Average Gender Diversity: 0.4936708860759494 Average Age Difference: 0.17721518987341772 Average Food Preference deviation: 0.0	Number of Pairs: 152 Number of Successors(Nachrückende): 6 Average Gender Diversity: 0.6282894736842105 Average Age Difference: 0.4342105263157895 Average Food Preference deviation: 0.0 ---Only those created by algorithm--- Number of Pairs created by algorithm: 79 Average Gender Diversity: 0.5 Average Age Difference: 0.3924050632911392 Average Food Preference deviation: 0.0
food2,age1,gender3:	**food3,age1,gender2:**
Number of Pairs: 154 Number of Successors(Nachrückende): 2 Average Gender Diversity: 0.6201298701298701 Average Age Difference: 0.24025974025974026 Average Food Preference deviation: 0.11688311688311688 ---Only those created by algorithm--- Number of Pairs created by algorithm: 81 Average Gender Diversity: 0.4876543209876543 Average Age Difference: 0.024691358024691357 Average Food Preference deviation: 0.2222222222222222	Number of Pairs: 153 Number of Successors(Nachrückende): 4 Average Gender Diversity: 0.6209150326797386 Average Age Difference: 0.23529411764705882 Average Food Preference deviation: 0.28104575163398693 ---Only those created by algorithm--- Number of Pairs created by algorithm: 80 Average Gender Diversity: 0.4875 Average Age Difference: 0.0125 Average Food Preference deviation: 0.5375
food2,age3,gender1:	**food3,age2,gender1:**
Number of Pairs: 151 Number of Successors(Nachrückende): 8 Average Gender Diversity: 0.6324503311258278 Average Age Difference: 0.5894039735099338 Average Food Preference deviation: 0.4370860927152318 ---Only those created by algorithm--- Number of Pairs created by algorithm: 78 Average Gender Diversity: 0.5064102564102564 Average Age Difference: 0.6923076923076923 Average Food Preference deviation: 0.8461538461538461	Number of Pairs: 151 Number of Successors(Nachrückende): 8 Average Gender Diversity: 0.6291390728476821 Average Age Difference: 0.5894039735099338 Average Food Preference deviation: 0.48344370860927155 ---Only those created by algorithm--- Number of Pairs created by algorithm: 78 Average Gender Diversity: 0.5 Average Age Difference: 0.6923076923076923 Average Food Preference deviation: 0.9358974358974359

Gruppenbildung

Für die Gruppenbildung ist createMealGroups(Course course) unsere Hauptmethode. Die Methode übernimmt einen Gang, für den eine Liste von Gruppen erstellt werden soll. Die erste Hilfsmethode, die wir benutzen, ist findPairsWithoutGroups(Course course).

Diese Methode nimmt einen Gang und gibt eine Liste von Paaren aus, die für den bestimmten Gang noch keiner Gruppe zugehören. Wir vermeiden damit, dass ein Paar für einen Gang zweimal in eine Gruppe zugewiesen wird. Wir nennen diese Liste von Paaren availablePairs.

Zunächst nehmen wir das erste Paar von dieser Liste und mithilfe der Methode findValidCandidatesForGroups(Pair pair, List availablePairs) finden wir eine Liste von Paaren, die mit unserem Paar kompatibel sind. Hier wird kontrolliert, ob die Essenspräferenzen der Paare geeignet sind, ob die Paare sich schon einmal gesehen haben, ob die Küchen dieser Paare nicht identisch sind, dass die Küche des Paars nicht mehr als dreimal benutzt wird usw. Danach wählen wir greedy das erste Paar, das zu den Anforderungen passt, bis die Gruppengröße genau 3 ist.

Nach diesem Schritt wird überprüft, ob es mindestens ein Paar gibt, das noch nicht gekocht hat und ob es mindestens eine Küche gibt, die frei für diesen Gang ist. Wenn ja wird diese Gruppe zu MealGroups Liste hinzugefügt. Wenn die Gruppen für alle Gänge bereit sind, werden diese Gruppen in eine Liste von allen Gruppen hinzugefügt.

Es wird danach mithilfe der Methode findKitchenOwnerPairInGroup
bestimmt, welches Paar in der Gruppe kochen soll. Wenn diese Gruppe eine Dessert-Gruppe ist, wird das Paar mit der Küche, die am nähsten zur Party-Location ist, gewählt. Wenn die Gruppe eine Appetizer-Gruppe ist, dann das Gegenteil davon. Somit nähern sich die Paare immer einen Schritt weiter der After-Party.