Willikins-Seating-Problem

Tom Kastek, Maximilian Wutz, Phil Sehlmeyer

Implementation

- Funktion: Solve*([Namensliste],[Relationsliste])
- Sortieralgorithmen: Brute Force, Greedy Ascent, erweiterter Greedy Ascent mit Random Walk und Random Restart

Fileservice and Randomize

- Die Klasse Fileservice kann Relationen aus einer Datei laden und/oder in eine Datei speichern
- Die Relationen können jedoch auch random anhand einer Personenliste generiert werden

Klasse: Relations und Permuter

Relations:

- ArrayList<String> people
- Hashtable<String,Integer> relations

Permuter:

- Generiert für eine ArrayList von Personen sämtliche Permutationen
- Generiert für eine ArrayList von Personen sämtliche Permutationen durch vertauschen

Brute Force

- Generiert durch den Permuter alle Permutationen von Sitzordnungen

- Bewertet jede ArrayList mit einem Rating - alle EinzelRatings zusammenaddiert

- Ist eine Sitzordnung besser als eine andere wird sie gespeichert und mit den folgenden verglichen

- Gibt die ArrayList mit dem höchsten Rating und das dazugehörige Rating aus

Greedy Ascent

- Überprüft Nachbarn auf bessere Ratings und legt eine neue Sitzordnung fest

- Falls keine bessere Ordnung gefunden werden kann terminiert der Algorithmus

- Falls doch, rekursiver Aufruf mit neuer bester Ordnung und neuer Berechnung der Permutationen

Greedy - erweitert

- Erweiterung des einfachen Greedy Algorithmus
- Soll verbinden das man in eine Sackgasse bzw. Schleife läuft
- Random Walk und Random Restart