BUILDING A WEDDING SEATING PLAN

Group_A1_127

- Gonçalo Magalhães
- Leonardo Teixeira
- David Carvalho

Specification of the work to be performed

Objective

The objective of a wedding seating plan optimization challenge is to place guests at tables in a way that maximizes pleasure while adhering to limitations. The issue encompasses a number of elements, including table size and relationships

Definition of the Problem

A set of tables with a set capacity and a set of guests are provided to you. The goal is to optimally assign guests to tables according to predetermined criteria

Limitations

- Table Capacity
- Guest Preferences
- Guest Conflicts
- Equitable Distribution

Objective Functions (Optimization Goals)

- Maximize guest satisfaction by seating compatible individuals together
- Minimize conflicts by ensuring that incompatible guests are seated at different tables

Method for Resolving the Issue

- Graph-based Algorithms: Represent guests as nodes and the connections between them as edges
- Constraint Satisfaction Problem (CSP): Apply constraint solvers or backtracking
- Metaheuristic Algorithms: Use optimization techniques such as simulated annealing, genetic algorithms, and other heuristic methods

Formulation of the problem

The Wedding Seating Plan problem assigns guests to tables while satisfying constraints and maximizing satisfaction. A solution maps guests to tables, respecting capacity limits and avoiding conflicts. Hard constraints must be met, while soft constraints, like seating friends together, are optimized. The problem is solved using search-based methods such as Genetic Algorithm, Greedy Algorithm, Tabu Search, Hill Climbing, K-Means Clustering or Simulated Annealing to maximize guest satisfaction.

Search Problem Formulation

- State Representation: A seating arrangement where guests are assigned to tables.
- Initial State: All guests unassigned
- Goal State: All guests assigned while satisfying constraints (e.g., no conflicts, ages, interests)
- Operators: Create a matrix table with a score for each guest relationship, and assign them based on that
- Heuristic: Maximizing satisfied preferences and minimizing conflicts

Optimization Problem Formulation

- Solution Representation: A list mapping guests to tables
- Mutation/Crossover: Random moves for mutation; mix table assignments for crossover.
- Hard Constraints: Table capacity, avoid conflicts
- Soft Constraints: Seat friends together, balance tables
- Evaluation Function: Score based on satisfied constraints, conflicts, and table balance

The Approach

Our approach focuses on optimizing table seating arrangements by maximizing guest happiness. We use heuristics to guide guest assignments based on relationships and balance tables efficiently. A well-defined evaluation function calculates total happiness by summing pairwise relationship scores. To improve seating plans, we apply neighborhood operators, such as swapping, shuffling, or moving guests between tables. Different optimization algorithms are used, including Greedy methods, Simulated Annealing, Genetic Algorithms, and Local Search, ensuring high-quality solutions. Advanced features like KMeans Clustering help pre-group guests, while progress tracking and numpy-based computations ensure efficiency.

1. Heuristics

- Rules for finding good solutions efficiently
- Guides guest assignments to maximize happiness
- Prioritizes strong relationships and balances tables

2. Evaluation Function

- Measures how good a seating arrangement is
- Calculates total happiness by summing pairwise scores
- Higher scores indicate better seating plans

3. Neighborhood Operators

- Modify the current solution slightly
- Swap guests between tables
- Shuffle guests within a table
- Move a guest to another table

4. Optimization Algorithms

- Greedy: Assigns guests based on heuristics
- Simulated Annealing: Allows worse solutions temporarily
- Genetic Algorithms: Uses evolution techniques
- Local Search: Iteratively improves seating plans (Hill Climbing, Tabu Search)

5. Advanced Features

- KMeans Clustering: Identifies groups of guests with similar relationships.
- Progress Tracking: Displays computation progress
- Numpy: Efficiently processes relationship scores
- These techniques optimize seating for maximum happiness

Implemented Algorithms

Greedy Algorithm	Simulated Annealing	Genetic Algorithm
• The greedy approach makes a series of local, optimal choices in the hope that they lead to a globally optimal solution. • In the context of table_optimizer.py, it likely selects the best immediate assignment or configuration based on a predefined criterion (e.g., minimizing cost, maximizing efficiency).	• Simulated Annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. • It starts with an initial solution and iteratively moves to a neighboring solution. • A worse solution may be accepted with a certain probability, which decreases as the algorithm progresses, helping to avoid local optima.	• GA is an evolutionary algorithm inspired by natural selection. • It starts with a population of candidate solutions (chromosomes), evaluates their fitness, and applies selection, crossover, and mutation operations to generate new candidates. • The process continues until a stopping criterion is met.

Implemented Algorithms

Hill Climbing	Tabu Search	K-Means Clustering
• Hill Climbing is an iterative algorithm that starts with an arbitrary solution and makes incremental changes. • It moves in the direction of increasing improvement. • If a better neighboring solution exists, it is accepted; otherwise, it stops when no better neighbors are found.	• Tabu Search (TS) is a metaheuristic that avoids cycling back to previously visited solutions. • It maintains a list of recently visited solutions (tabu list) to prevent the search from revisiting them. • It explores the search space even if the current move is not the best, avoiding local optima.	• K-Means is an unsupervised clustering algorithm that partitions data into k clusters. • It assigns each point to the nearest cluster center and iteratively updates cluster centers until convergence. • Used to group similar entities together, often as a preprocessing step in optimization problems.

Experimental Results

Dataset: Medium-sized database
Algorithms:

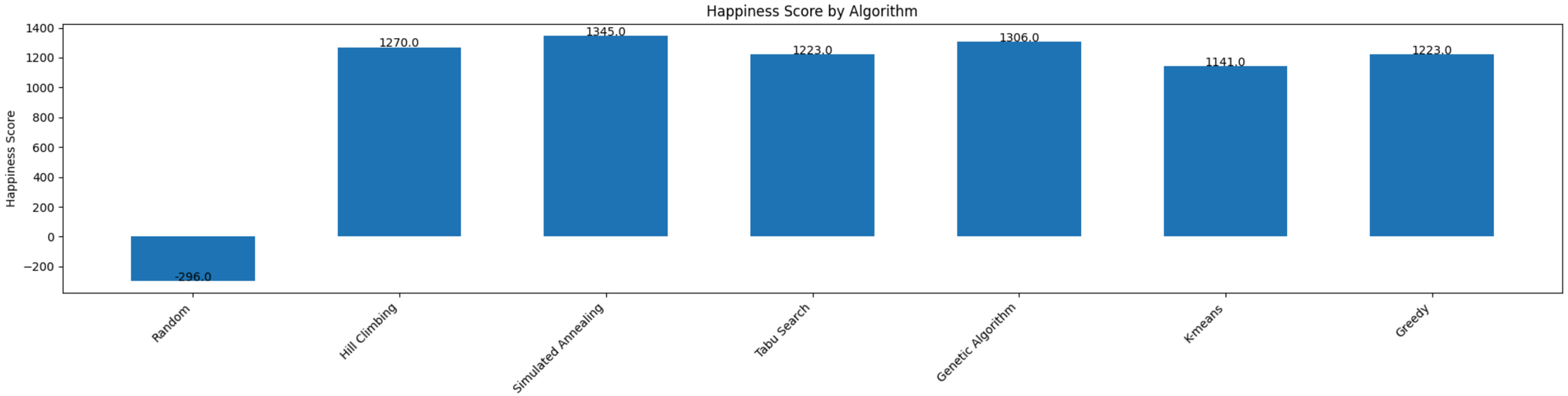
- Random Assignment
- Hill Climbing
- Simulated Annealing
- Tabu Search
- Genetic Algorithm
- K-Means Clustering
- Greedy Algorithm

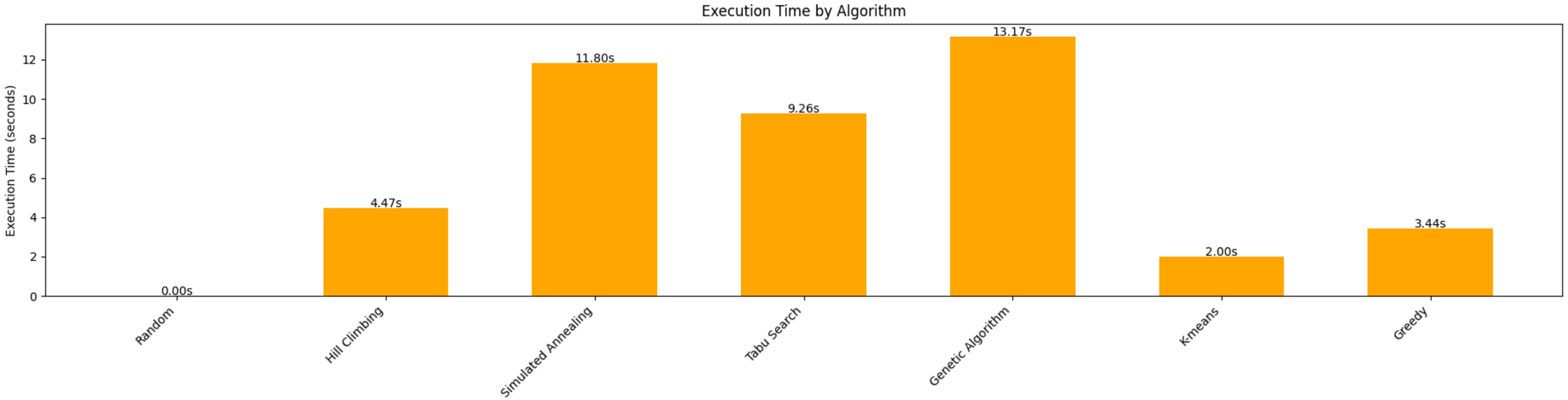
Metrics:

- **Happiness Score:** Measures the effectiveness of the solution
- **Execution Time:** Measures time
- **Software Used:** Python with Matplotlib for visualization
- **Efficiency:** Happiness Score / Second

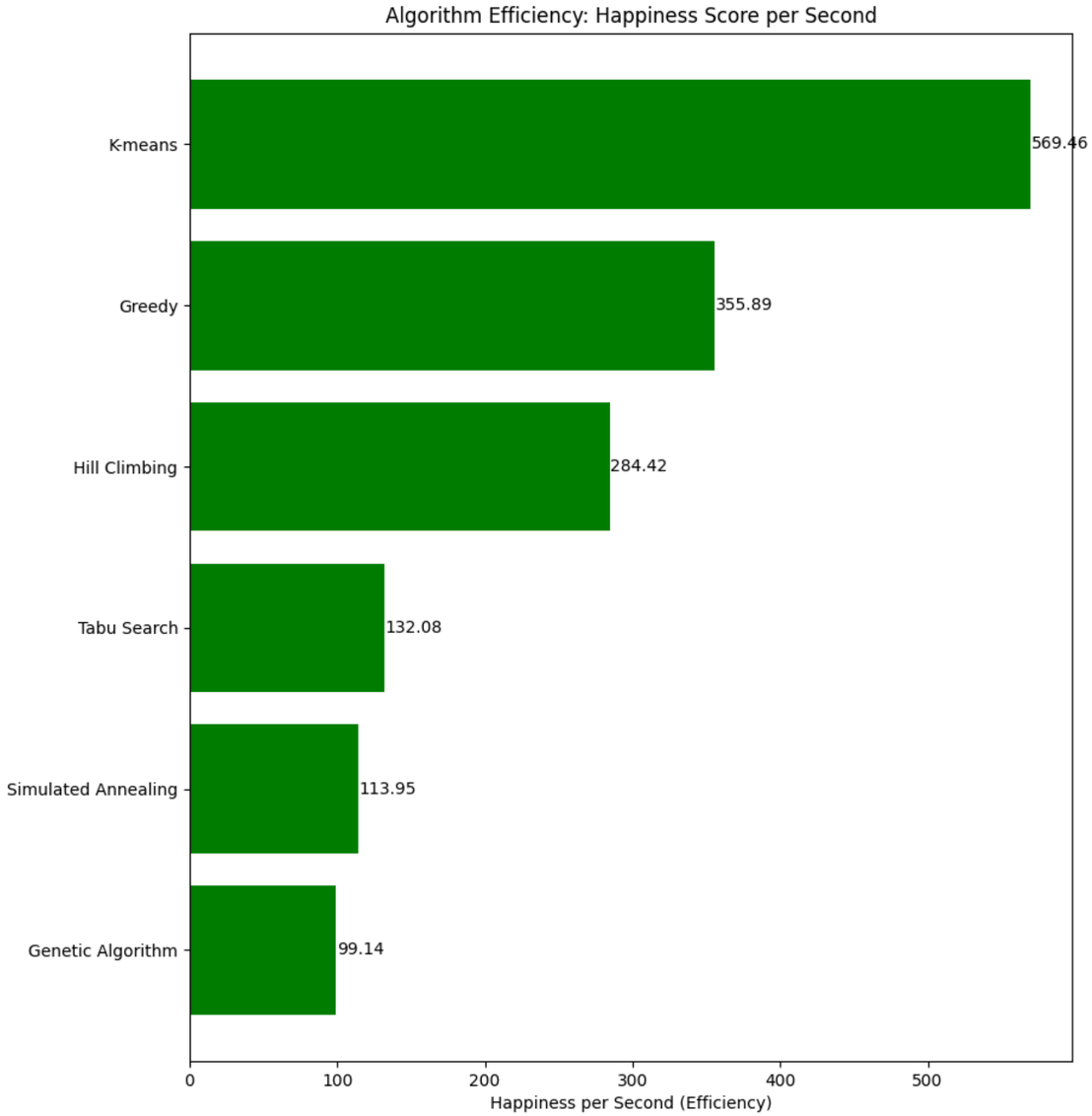
Algorithm	Happiness Score	Efficiency	Time (s)
Random	-296	---	0.0042
Hill Climbing	1270	284.42	4.4652
Simulated Annealing	1345	113.95	11.8034
Tabu Search	1223	132.08	9.2594
Genetic Algorithm	1306	99.14	13.1735
K-Means Clustering	1141	569.46	2.0036
Greedy Algorithm	1223	355.89	3.4364

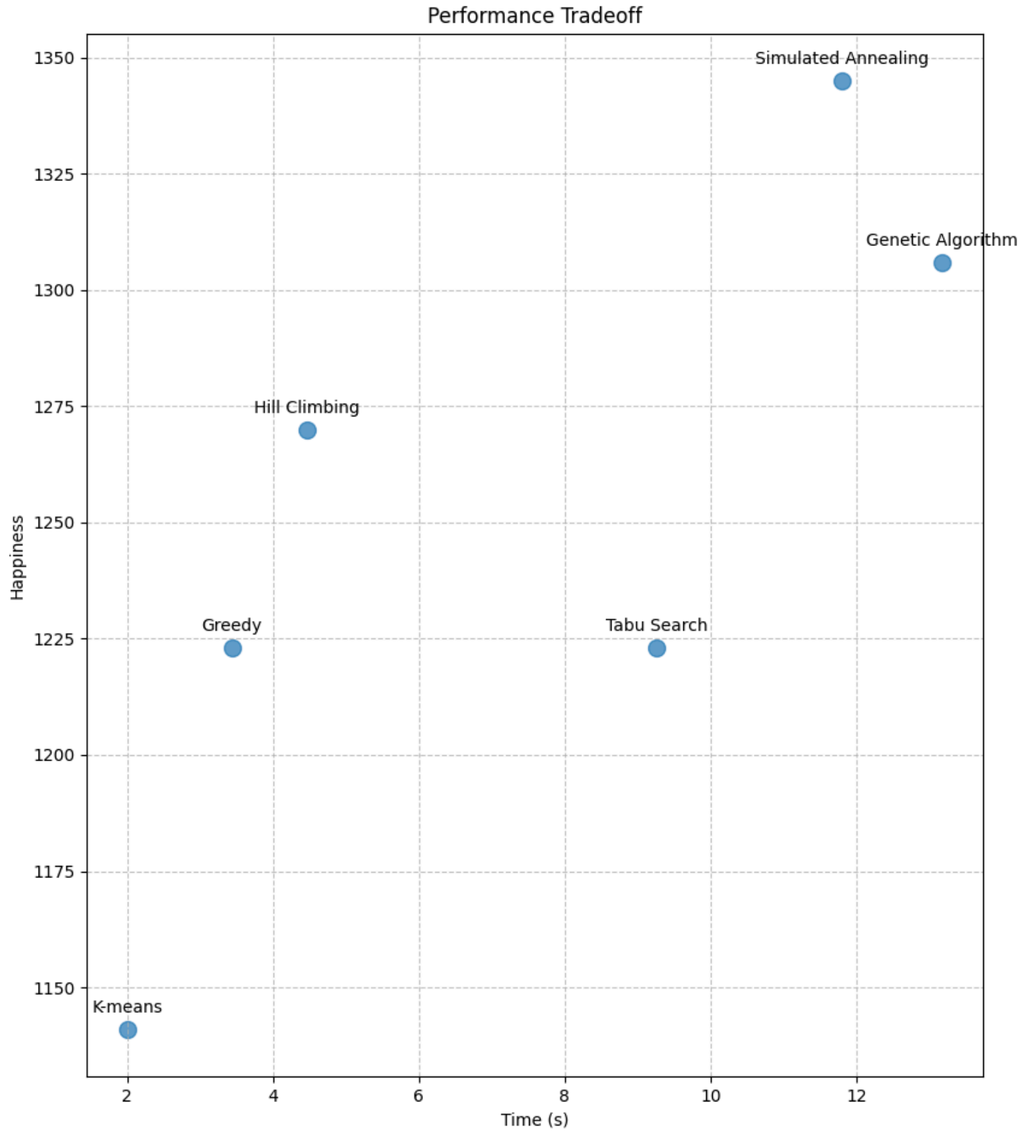
Charts


Happiness Score by Algorithm
Happiness Score
1400
1200
1000
800
600
400
200
0
−200
-296.0
1270.0
1345.0
1223.0
1306.0
1141.0
1223.0
Random
Hill Climbing
Simulated Annealing
Tabu Search
Genetic Algorithm
K-means
Greedy

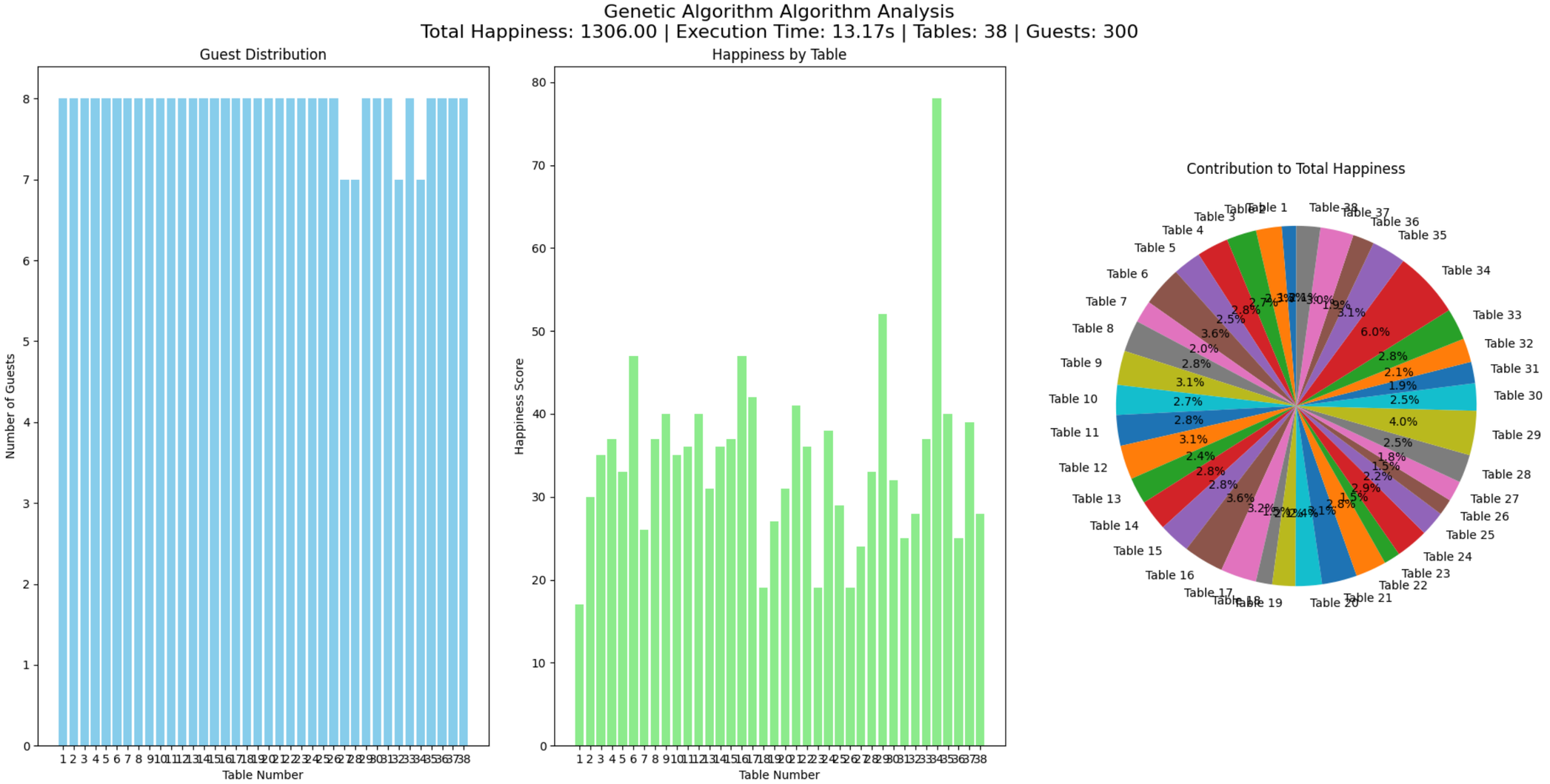

Execution Time by Algorithm
Execution Time (seconds)
12
10
8
6
4
2
0
0.00s
4.47s
11.80s
9.26s
13.17s
2.00s
3.44s
Random
Hill Climbing
Simulated Annealing
Tabu Search
Genetic Algorithm
K-means
Greedy

Charts


Algorithm Efficiency: Happiness Score per Second
K-means
Greedy
Hill Climbing
Tabu Search
Simulated Annealing
Genetic Algorithm
569.46
355.89
284.42
132.08
113.95
99.14
0
100
200
300
400
500
Happiness per Second (Efficiency)


Performance Tradeoff
Simulated Annealing
Genetic Algorithm
Hill Climbing
Greedy
Tabu Search
K-means
1350
1325
1300
1275
1250
1225
1200
1175
1150
Happiness
2
4
6
8
10
12
Time (s)

Charts


Genetic Algorithm Algorithm Analysis
Total Happiness: 1306.00 | Execution Time: 13.17s | Tables: 38 | Guests: 300
Guest Distribution
Number of Guests
Table Number
Happiness by Table
Happiness Score
Table Number
Contribution to Total Happiness
Table 1
Table 2
Table 3
Table 4
Table 5
Table 6
Table 7
Table 8
Table 9
Table 10
Table 11
Table 12
Table 13
Table 14
Table 15
Table 16
Table 17
Table 18
Table 19
Table 20
Table 21
Table 22
Table 23
Table 24
Table 25
Table 26
Table 27
Table 28
Table 29
Table 30
Table 31
Table 32
Table 33
Table 34
Table 35
Table 36
Table 37
Table 38

Conclusion

SUMMARY

- Most Effective Algorithm: Simulated Annealing with reheating provides the best overall results, with 10-20% improvement over greedy solutions
- Best Time/Quality Tradeoff: Tabu Search offers excellent solution quality with moderate runtime requirements
- For Large Problems: K-means clustering provides good initial solutions that can be refined by metaheuristic algorithms
- Parameter Settings: Initial temperature and cooling rate are critical for Simulated Annealing success
- Hybrid Approach: Starting with greedy or K-means solutions and refining with metaheuristics yields the best results
- Implementation Insight: The handling of table balancing has significant impact on final solution quality

SOFTWARE AND TOOLS USED

- Python for implementation
- NumPy for efficient matrix operations
- Scikit-learn for K-means clustering implementation
- tqdm for progress tracking
- Matplotlib and seaborn for data visualization
- GitHub for version control and project management

REFERENCES

1. *A Heuristic Approach in Solving the Optimal Seating Chart Problem*
2. *Wedding Seating Problem Using Simulated Annealing*
3. *Optimizing a Round Table Seating Arrangement Using Algorithms*
4. *Using a Genetic Algorithm for Table Seating*
5. *Simulated Annealing Seating Chart*
6. *Round Table Seating Arrangement: Search and Heuristic Approaches*
7. *Building a Wedding Seating Plan Using Probabilistic Methods (Simulated Annealing)*