# Optimizing Santa's Workshop Tour Scheduling Using Genetic Algorithms

Adagamov Nurmukhammet, Bulat Fakhrutdinov, Danil Eramasov

***Abstract*—This project aims to solve the Santa's Workshop Tour 2019 Kaggle competition by optimizing the scheduling of 5,000 families to 100 tour dates using a Genetic Algorithm (GA). The goal is to minimize the total penalty cost, which includes preference penalties (based on how well the assigned day matches the family's preferences) and accounting penalties (based on daily attendance fluctuations). The proposed approach leverages problem-specific operators, local search, and efficient fitness evaluation to achieve a high-quality solution. The dataset includes family preferences and sizes, and the project will be implemented in Python, utilizing libraries such as NumPy and DEAP for GA implementation.**

***Index Terms*—Genetic Algorithm, Scheduling Optimization, Santa's Workshop, Penalty Minimization, Kaggle Competition**

## I. Introduction

The Santa's Workshop Tour 2019 Kaggle competition [1] presents a challenging scheduling problem where 5,000 families must be assigned to one of 100 tour dates. Each family has provided their top 10 preferred dates, and the goal is to minimize the total penalty cost, which includes both preference penalties and accounting penalties. Preference penalties arise when families are assigned to less preferred dates, while accounting penalties are incurred due to fluctuations in daily attendance. This project proposes a Genetic Algorithm (GA) approach [2] to solve this problem efficiently, balancing exploration and exploitation to find a near-optimal solution.

## II. Project Idea

The project involves developing a GA-based solution to assign each family to a tour date while minimizing the total penalty cost. The GA will use problem-specific operators, such as preference-biased mutation and accounting-aware crossover, to ensure that the solution respects both family preferences and daily attendance constraints. The fitness function will evaluate the total penalty, and local search techniques will be integrated to refine solutions further. The project will be implemented in Python, and the results will be submitted to the Kaggle competition for evaluation.

## III. Method/Technique

The proposed method is a Genetic Algorithm (GA) implemented in Python using the DEAP framework [3]. The components include:

- **Solution Representation**: Chromosome represented as an array where each index corresponds to a family, and the value represents the assigned tour date.
- **Fitness Function**: Combines preference penalties and accounting penalties to evaluate the quality of a solution.
- **Selection**: Tournament selection to choose parents for reproduction and ensure genetic diversity.
- **Crossover**: Uniform crossover with a bias towards better-performing parents.
- **Mutation**: Preference-biased mutation to explore better assignments (prioritizing large families).
- **Local Search**: Further optimization using problem-specific heuristics.
- **Elitism**: Retains the best solutions across generations.

## IV. Dataset Explanation

The dataset [4] consists of a single file, `family_data.csv`, which contains the following columns:

- `family_id`: Unique identifier for each family.
- `n_people`: Number of people in the family.
- `choice_0` to `choice_9`: The top 10 preferred dates for the family.

The dataset is available at: [4].

## V. Timeline

The project will be completed over four weeks with the following timeline and individual contributions:

- **Week 1**: Problem understanding and dataset analysis.
  - Adagamov Nurmukhammet: Analyze preference penalties and their impact on the fitness function.
  - Bulat Fakhrutdinov: Analyze accounting penalties and their mathematical formulation.
  - Danil Eramasov: Set up the Python environment and install necessary libraries (e.g., NumPy, DEAP).
- **Week 2**: Design and implement the Genetic Algorithm framework.
  - Adagamov Nurmukhammet: Implement solution representation and fitness function.
  - Bulat Fakhrutdinov: Implement selection and crossover operators.
  - Danil Eramasov: Implement mutation and local search components.
- **Week 3**: Initial implementation and testing.
  - Adagamov Nurmukhammet: Test the fitness function with sample data.
  - Bulat Fakhrutdinov: Validate crossover and selection operators.
  - Danil Eramasov: Test mutation and local search on small subsets of the dataset.

- **Week 4**: Optimization and parameter tuning.
  - Adagamov Nurmukhammet: Fine-tune GA parameters (e.g., population size, mutation rate).
  - Bulat Fakhrutdinov: Optimize the crossover operator for better performance.
  - Danil Eramasov: Improve local search efficiency and integrate it with the GA.
- **Week 5**: Large-scale testing and performance evaluation.
  - Adagamov Nurmukhammet: Run the GA on the full dataset and evaluate results.
  - Bulat Fakhrutdinov: Analyze the impact of accounting penalties on the final solution.
  - Danil Eramasov: Conduct parallelization tests to improve computational efficiency.
- **Week 6**: Final submission and report writing.
  - Adagamov Nurmukhammet: Write the methodology section of the report.
  - Bulat Fakhrutdinov: Write the results and discussion section.
  - Danil Eramasov: Prepare the final submission and Kaggle leaderboard entry.

## REFERENCES


[1] Kaggle Competition: Santa's Workshop Tour 2019. https://www.kaggle.com/competitions/santa-workshop-tour-2019/overview

[2] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

[3] DEAP: Distributed Evolutionary Algorithms in Python. https://deap.readthedocs.io/

[4] https://www.kaggle.com/competitions/santa-workshop-tour-2019/data