Optimizing İSPARK Parking Lot Management in Istanbul

Batuhan Sal
150210316
sal21@itu.edu.tr

Ömer Erdağ
150210332
erdag21@itu.edu.tr

Serdar Biçici
150210331
bicici21@itu.edu.tr

***Abstract*—A genetic algorithm is an optimization technique inspired by natural selection. It uses processes like selection, crossover, and mutation to evolve solutions to complex problems. Starting with a random population to iteratively improve solutions based on their fitness scores. Widely used in engineering, AI, bioinformatics and it excels in large, complex search spaces where traditional methods fall short. In this project, the genetic algorithm method is used for optimizing parking lot-car matchups. In a city like İstanbul, finding a parking stop is a rare opportunity however, this project aims to change this fact. By using a DNA sequence of parking lots, it optimizes which car should go to which parking lot by every generation. In addition to the classical approach, which is every car should go to the nearest parking space, this optimizing process takes more inputs such as budget, gas cost and parking cost.**

***Index Terms*—Genetic algorithm, car, park, optimization**

I. Problem Description

First of all, we applied some changes to our proposal. We were aiming to use dynamic pricing strategies depending on real-time demand and also maximize revenue for İSPARK. However, we thought that it would not be very suitable to implement. Thus, we aim to minimize the total cost spent by the customer and maximize space utilization and user satisfaction in our project.

The main problem is to assign each car a perfect match of parking lot space in İstanbul depending on some parameters. The classical approach to this problem is assigning the closest parking lot to each car, however, this may cause overcrowded parking spaces and inefficacy in the big picture. Just rotating to another parking lot will increase CO_2 emissions and disrupt the daily life of the individuals. This project's approach is the expanded version of the classical approach. Still, the distance between the car and parking space is of high importance but with the parameters of gas cost, budget and parking cost added, the problem and the approach becomes much deeper.

II. Method Formulation

In this part firstly, we will explain our code structure and how we define functions and classes to solve our problem, secondly we will introduce our fitness functions for the genetic algorithm proccess.

Dataset and Preprocessing

Firstly, we got the parking lot data from the Ispark Api with request from Park. Then we get the id's from the parking lot information which is saved on the CSV file. After that, we use Sample Id for price information, capacity, and location. We saved the parking data information to a data frame. In the data frame, we preprocessed the data because there are some bus stops, wholesale markets and forests.

Park Class

We developed a Park class that can handle various functionalities including retrieving the park's ID, location, capacity, and capacity rate. Additionally, it allows for adding and removing cars from the parking lot, checking if the lot is empty or not, and providing price information based on the given hours. With this class, we created our parking lots.

Car Class

For the cars, we developed a class which can get the location, gas usage and parking time in hour format. For convenience, we obtained data from bus stop coordinates in Istanbul and sampled 100 cars from these locations.

Genetic Algorithm for Car-Parking Space Allocation

The genetic algorithm function optimizes the assignment of cars to parking lots using a genetic algorithm. It includes several helper functions:

- **fitness function**: Evaluates the fitness of a given assignment by calculating the total cost (including gas and parking costs), space utilization, and user satisfaction. It adjusts these metrics based on whether the parking lot is empty or over capacity, and the specified mode.
- **create initial population**: Generates an initial population of random assignments of cars to parking lots.
- **selection**: Selects the top half of the population based on their fitness scores to act as parents for the next generation.
- **crossover**: Combines two parent assignments to produce offspring by exchanging segments of their assignments.
- **mutate**: Randomly alters an assignment with a certain probability to introduce variation.

The main genetic algorithm function coordinates the process over a number of generations, evolving the population by repeatedly selecting, crossing over, and mutating individuals. It tracks the best assignment found and its fitness score. The function concludes by returning the best assignment, its fitness score, and the fitness scores over generations.

In summary, this function employs a genetic algorithm to find the optimal assignment of cars to parking lots by iteratively improving the population of potential solutions through natural selection-inspired mechanisms. After that, we plotted the results and we compared them with another fitness function approach.

Fitness Functions

We use 2 different fitness functions to evaluate our parking management. x is the total cost which consists of gas price, park price, and penalty due to full spot. y is space utilization which is the ratio of the empty capacity. And we introduce a new variable with z which is user satisfaction. This variable incremented when we placed the car in an empty spot and decremented when we placed the car in a full spot.

Reverse Linear:

$$f(x, y, z) = \frac{1}{0.8 \cdot x - 0.1 \cdot y - 0.1 \cdot z}$$

Linear:

$$f(x, y, z) = -0.8 \cdot x + 0.1 \cdot y + 0.1 \cdot z$$

In both functions, we aim to return maximum values.

III. Real-World Applications

This project's real-world applications are of high importance to metropolises of the world. Both private and government sectors can use the outputs of this project to optimize their whole fleets of vehicles to park or dock more efficiently. Taxi service providers, motor couriers, bus fleets, e-scooter fleets and countless other vehicles can use this service to optimize their parking and help reduce environmental risks of high emissions, noise pollution and disruption of daily life.

IV. Experimental Evaluation

Base Experiment

The base experiment is assigning every car to its closest parking lot without taking any account of other parameters. This will be used for the evaluation of the genetic algorithm. In our test run, the result is this:

```
Average distance between every car and nearest
    parking lots: 1.6466484295392114 km
```

Experiment No.1 Population Size

As shown in the figure, there is a positive correlation between population size and fitness score. With a larger population, the chance of much better solutions is increasing and therefore giving better outputs.

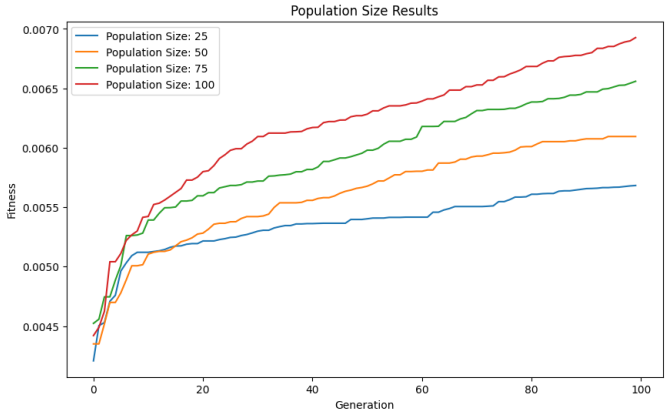

Fig. 1. Experiment No.1 Population Size

Experiment No.2 Mutation Rate

When the genetic algorithm is tried with different mutation chances, it gives different fitness scores within the same number of generations. As can be deduced from the graph, with higher mutation chances the fitness scores become higher. However, in the last generations, the 0.2 mutation rate is nearly kept up with the 0.3 mutation rate.

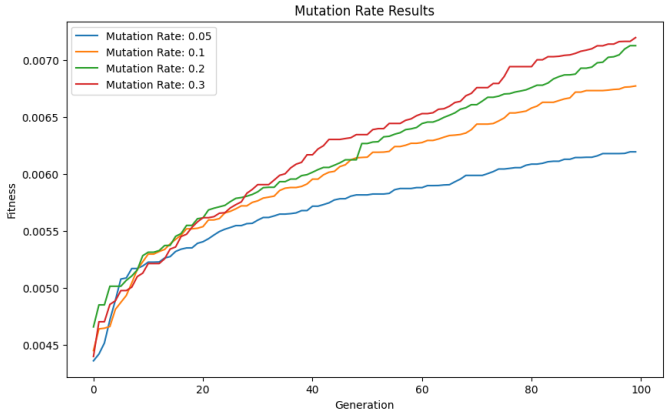

Fig. 2. Experiment No.2 Mutation Rate

Experiment No.3 Generation Number

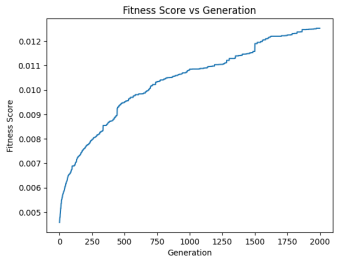

Fig. 3. Experiment No.3 Generation Number

In this experiment, we have tried to find if a great number of generations are necessary for better fitness scores. However,

as can be seen from the graph, the increase of the function is decreasing, therefore it is slowly converging and great numbers of generations are not necessary to compute.

Experiment No.4 Different Fitness Functions

When we experiment with both reverse linear and linear fitness functions, we get similar but slightly different outputs. All the experiments done until here were being done by the reverse linear fitness function and now we are going to compare them.

```
# 1st fitness function
Average distance between cars and parking lots:
    4.86991603136533 km
Average price paid by cars: 113.82042184069624 TL
```

```
# 2nd fitness function
Average distance between cars and parking lots:
    4.433264020262411 km
Average price paid by cars: 115.91063893256882 TL
```

The first fitness function starts from the value of $0.0042...$ and after 1000 generations converges to $0.012...$ while the second fitness function starts from $-232.93...$ and finishes with $-78.94....$ Finally, we can declare our 2^{nd} function is the better one even if there is a small price gap between them. However, the 400 meter difference is what makes the n^{nd} fitness function better.

Genetic Algorithm & Nearest Parking Lot Strategy

Finally, we compare our optimized genetic algorithm (with 2^{nd} fitness function) to the strategy of parking every car at its nearest parking lot. The numerical outputs are like this:

```
# base strategy
Average distance between cars and parking lots:
    1.6466484295392114 km
Average price paid by cars: 128.3676108958862 TL
```

```
# genetic algorithm with 2nd fitness function
Average distance between cars and parking lots:
    4.433264020262411 km
Average price paid by cars: 115.91063893256882 TL
```

At first sight, the base strategy may seem better with a little price trade-off. However, there is a different aspect of comparison.

```
# base strategy
Number of cars assigned to full parking lots: 24
```

```
# genetic algorithm with 2nd fitness function
Number of cars assigned to full parking lots: 0
```

Finally, the superiority of our optimization emerges. While matching up the cars and parking spots, we take into consideration the empty spaces as well unlike the base strategy. The little difference between the nearest parking lot and the recommended parking lot may seem intolerable but it can easily tolerate the rerouting to another spot cost of the vehicle.

V. Conclusion

In conclusion, this project demonstrates the efficiency of using genetic algorithms for optimizing car-parking lot assignments in complex urban environments like Istanbul. By incorporating parameters such as gas cost, budget and parking fees, the genetic algorithm surpasses the classical nearest-parking strategy in minimizing overall costs and avoiding overcrowded parking lots. Through various experiments, we observed that increasing the population size and mutation rate generally enhances the fitness score, while an excessive number of generations yields diminishing returns. The comparison between linear and reverse linear fitness functions showed that while both are effective, the linear function offered marginally better performance in terms of distance and cost metrics. Importantly, the genetic algorithm achieved a balanced distribution of cars, ensuring no parking lot was over capacity, unlike the nearest-parking strategy, which led to overcrowding. These findings highlight the potential of genetic algorithms to improve urban parking management, leading to reduced emissions, lower costs for users and more efficient use of parking resources.