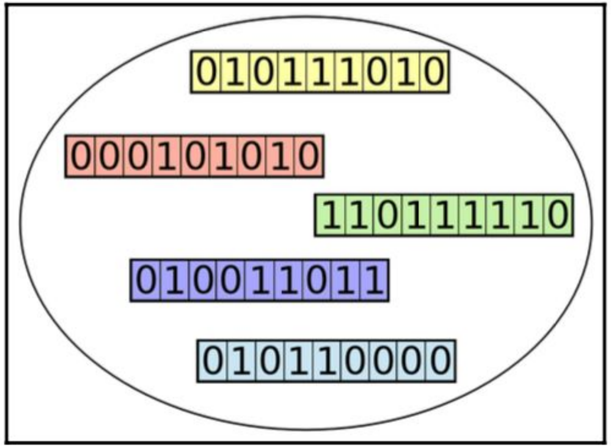
Topic 9: Optimize price to handle request

Alexander Tampier, Jan Köck, Milena Meier, Sofiia Badera

universität wien

Genetic Algorithms

- Inspired by the theory of natural selection
- Principle: Treat solutions as individuals in a population
    - Set of parameters → genes
- Fitness
    - effectiveness in solving the problem
- Strengths:
    - Technique is robust
        - wide range of difficult problems
    - Acceptably good solution in a reasonably quick time
    - Mechanism is robust
        - Degree of flexibility in parameter settings
- Weaknesses:
    - Risk of converging on a local maximum

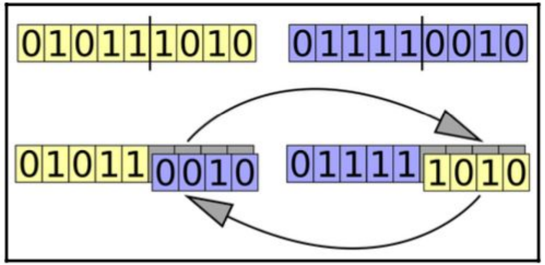


A population of individuals representated by binary-coded chromosed
Source: E. Wirsansky, *Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing Ltd, 2020.

Mechanisms of Genetic Algorithms

Simulate the different mechanisms of natural selection and evaluation

- Selection
  - Determines how individuals are chosen for reproduction
  - Selected based on fitness
- Crossover
  - Genetic material from two parent individuals is combined to produce offspring
- Mutation
  - After Crossover to each offspring
  - Introduces random changes in the genetic information
  - Maintains genetic diversity in population



Crossover operation between two binary-coded chromosomes
Source: https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.One.Point.svg.Image by Yearofthedragon. Licensed under Creative Commons CC BY-SA 3.0: https://creativecommons.org/licenses/by-sa/3.0/deed.en

Shortly the problem / issue to solve

- Implementing GA to develop optimal pricing strategy
- Defining optimization goals
  - Optimize for company profit
  - Optimize for customer price
  - Optimize for customer satisfaction
- Evolving factors like customer behavior and external influences
  - Customer loyalty
  - Weather
  - Time/day
  - Geographic and demographic factors

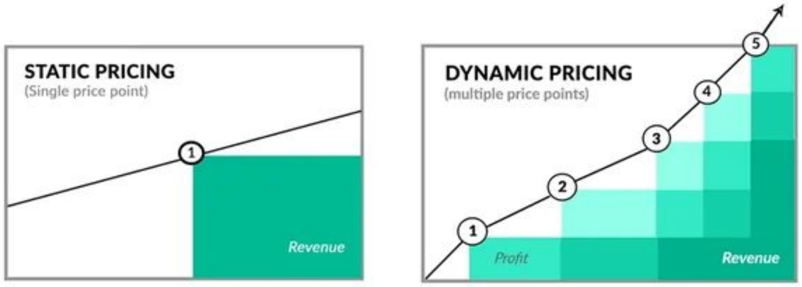


Figure 1: Static Pricing vs. Dynamic Pricing
Source from HubSpot Blog

Company Profit Optimization

- Dynamic Parameters
  - Customer loyalty, Weather conditions, Day of the week, Remoteness
- Static Constants
  - Baseline prices, Loyalty Surcharges, Weather Surcharges, Weekend Surcharges, Remoteness Surcharges
- GA Process
  - Selection, crossover, and mutation operations
- Optimization
  - Runs through multiple generations to improve the pricing strategy
  - Bounds are set to keep the adjustments within realistic limits
- Result → Optimized price considering all variables

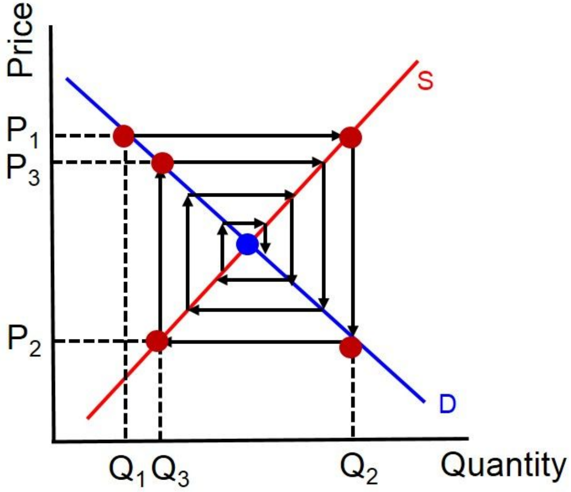


Figure 2: Influenced by Cobweb Model Price Convergence
Source from ezyeducation.co.uk

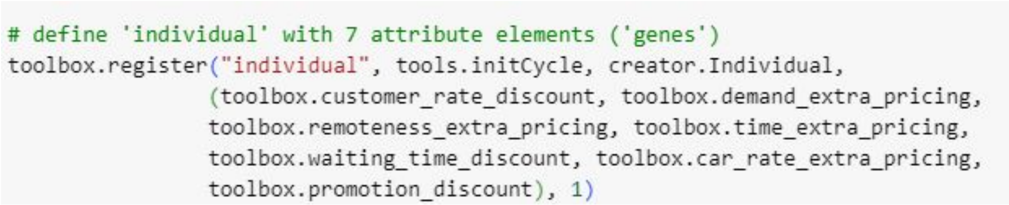
Customer Price Optimization

- Finding a solution for an optimal price for one ride
- Parameters
  - Dynamic basic tariff
    - Based on the time of day
  - Vehicle class factor
    - Standard - Premium - Luxury
  - Cost per km
  - Distance of the ride
    - Start - Destination
  - Remotnesse
  - Loyalty discount
    - Discounts based on the total distance traveled

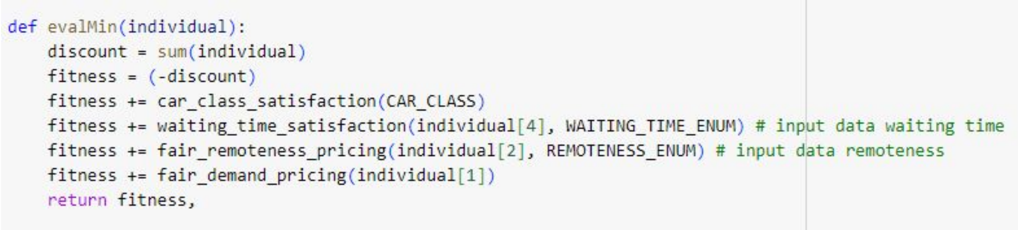
- GA Process
  - Based on min Problem
  - Selection, crossover, and mutation operations
- Fitness Function
  - Valuation based on the total cost calculation
- Optimization
  - Execution of the genetic algorithm over several generations
- Result
  - Minimal costs for one ride from start to destination

Customer Satisfaction Optimization

- Input Parameters
  - Customer Rate, Car Class, Base Price, Car / Customer / Destination Locations (Addresses)
- Real Time Parameters
  - Date, Number of events, Distance / Time calculations from Google Maps
- Fitness Function
  - Based on price, car class, waiting time and balanced pricing percent
- Result → Individual with best pricing strategies



```
# define 'individual' with 7 attribute elements ('genes')
toolbox.register("individual", tools.initCycle, creator.Individual,
                 (toolbox.customer_rate_discount, toolbox.demand_extra_pricing,
                 toolbox.remoteness_extra_pricing, toolbox.time_extra_pricing,
                 toolbox.waiting_time_discount, toolbox.car_rate_extra_pricing,
                 toolbox.promotion_discount), 1)
```



```
def evalMin(individual):
    discount = sum(individual)
    fitness = (-discount)
    fitness += car_class_satisfaction(CAR_CLASS)
    fitness += waiting_time_satisfaction(individual[4], WAITING_TIME_ENUM) # input data waiting time
    fitness += fair_remoteness_pricing(individual[2], REMOTENESS_ENUM) # input data remoteness
    fitness += fair_demand_pricing(individual[1])
    return fitness,
```

Live Demonstration

Evaluation of the solution

Challenges

- Ensuring realistic prices through the genetic algorithm
    - Use of max(individual) in the evaluate function to limit the parameters
    - Subsequent solution: checkBounds for offsprings
        - Monitoring and adaptation of each gene in each individual in the offspring
        - More effective method of limiting the parameter values