Vehicle Routing Problem

Group 5 :
Ann-Christin Meisener, Jannik Zeiser, Caro Gaß, Kim Gerbaulet, Adrian Rojas , Shahd

Overview

Constructing Phase

Initialization
Ant Colony Optimization

Improving Phase

Ant Colony Optimization

Initialization

Aim:

- distribute customers to cars so that all customer demands are covered
  - car type is chosen randomly
  - city is chosen randomly
  - only valid solutions are created
  - in case one car cannot fulfill a customer’s demands (anymore) another car will visit additionally

Initialization - Variations

To improve our algorithm we change the Initialization :

- when customers were assigned, the nearest customer can be chosen and not only a random customer +
- in a mode car classes could be set: either random or each of the classes (to see which might be the best) +
- the first customer of the next car will be close to the customers of the last car -

Initialization ... using classes

```
class Car(object):

    def __init__(self, capacity, cost, route, route_cost):

        self.capacity = capacity

        self.cost = cost

        self.route = route

        self.route_cost = route_cost

    def update_route_and_route_cost(self, route, route_cost):

        self.route = route

        self.route_cost = route_cost
```

Initialization -- Original

```
while not all customer demands are fulfilled:

      randomly choose a car

      while car capacity is > 0:

            randomly choose customer

            if customer demand < leftover car capacity:
                  set customer demand to 0
                  reduce car capacity by customer demand
            else:
                  set car capacity to 0
                  reduce customer demand by car capacity
```

Initialization -- New Version

```
while not all customer demands are fulfilled:

      randomly choose a car (or assign a car set)

      while car capacity is > 0:

            choose nearest customer

            if customer demand < leftover car capacity:
                  set customer demand to 0
                  reduce car capacity by customer demand
            else:
                  set car capacity to 0
                  reduce customer demand by car capacity
```

Another possible solution representations

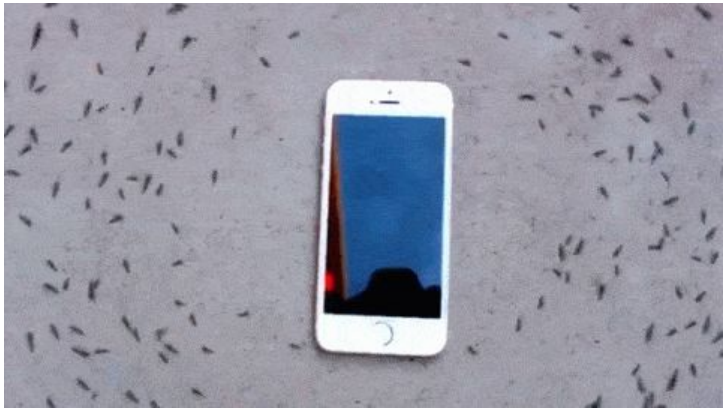
A solution could also be a list of cars and a list of customers.

- evaluating a solution would assign customers to cars given some rules
- GA could work on mutating and cross-over between list of cars and list of customers separately.

Ant Colony Optimization

Aim:

- After cars have been assigned to customers, find the shortest route for each of the cars using the ACO
  - Build on our existing ACO-implementation and adjust it to the problem



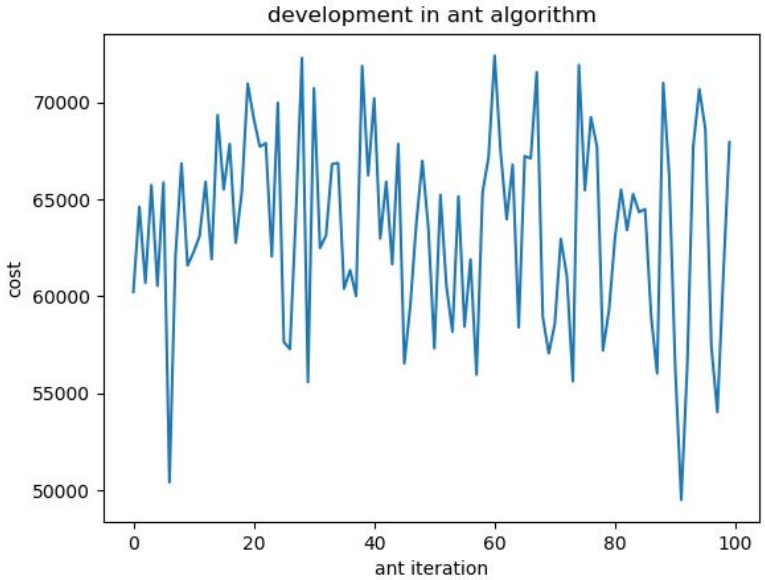
ACO

ACO pseudocode

```
for each car in solution:
     get customer list
     create pheromone matrix with defined default values
     create heuristic matrix
     for number of iterations:
          set start and end to base-station
          pheromone evaporation
          for each ant in colony:
               choose starting customer from customer list
               choose next customer from list according to probability
               function (roulette wheel) until solution is complete
          pheromone intensification
     save best route
```

Results New

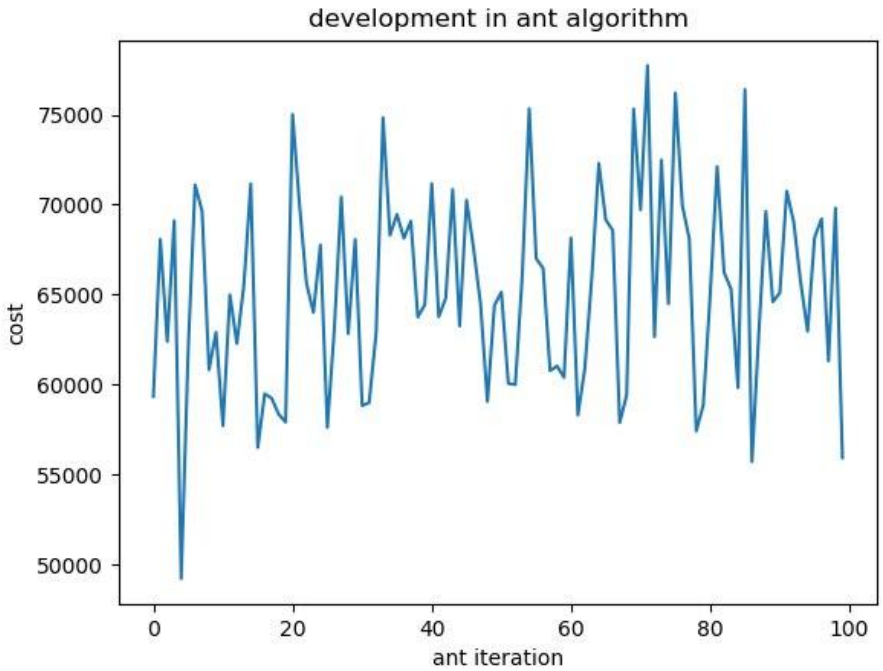
Parameters:
population size: 100
number of ants: 50
iterations: 50
car choosing mode: random
alpha: 1
beta: 0.5
delta: 0.1
evaporation parameter: 0.2
intensification mode: all-ants
path choosing mode: max
minimal cost (best solution): 49498



Results new

Parameters

population size: 100
number of ants: 20
car choosing mode: random
alpha: 1
beta: 1
delta: 0.1
path-choosing mode: roulette wheel
minimal cost (best solution): 49205



Results New

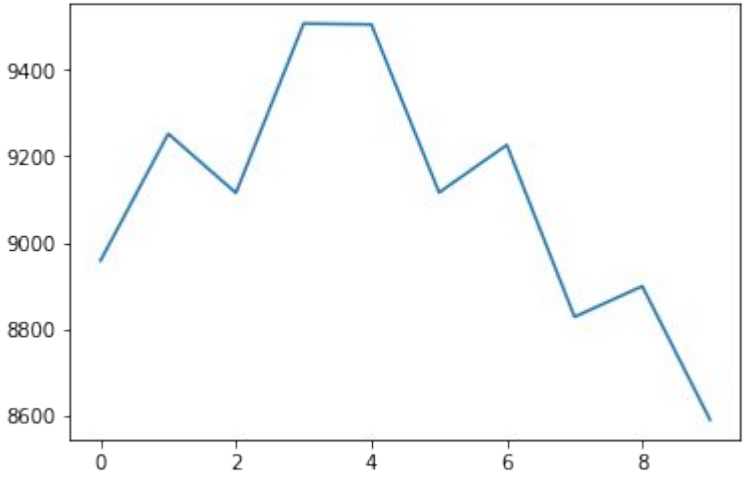
Parameters:
Colony size: 20
Iteration: 50
ph_value: 0.5
ev_parameter: 0.2
delta: 0.1
alpha: 1
beta: 1
pop _size: 100

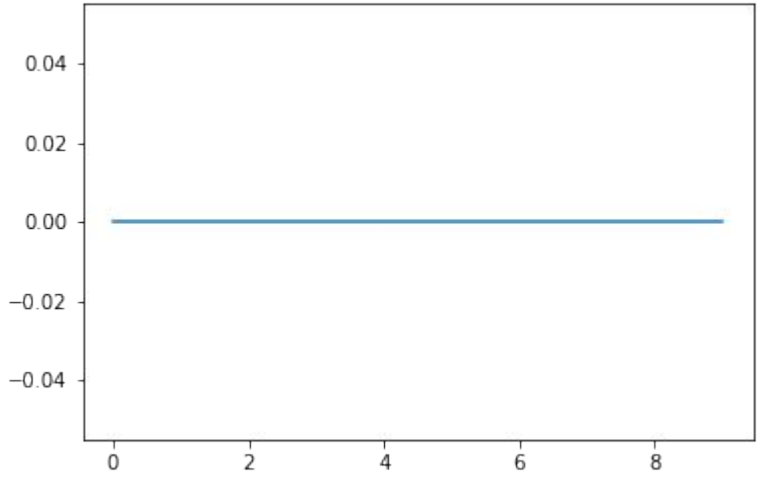
Roulette Wheel

Different car types	Results
low	58032
low-middle	50895
high-middle	56055
high	66132
random	49205

Results -- Old

Example:





ACO for two different cars. A used one and an used one

ant number = 50
iterations = 10
ph_value = 0.5
beta = 0.5
ev_parameter = 0.2
delta = 0.1
aplpha = 1

Results

ACO parameters:
ant number = 50
ev_parameter = 0.2
iterations = 10
delta = 0.1
ph_value = 0.5
alpha = 1
beta = 0.5

GE parameters:
roulette wheel
uniform crossover
mutator = random resetting
population size = 50
crossover rate = 1
mutation rate = 0.4

Development over Generations:

Generation 1-7:	236499
Generation: 8 - 38:	188024

unchanging number is due to printing only the best / better solutions

Results Compared

Old Results

ACO parameters:
ant number = 50
ev_parameter = 0.2
iterations = 10
delta = 0.1
ph_value = 0.5
alpha = 1
beta = 0.5

GE parameters:
roulette wheel
uniform crossover
mutator = random resetting
population size = 50
crossover rate = 1
mutation rate = 0.4

Generation 1-7: 236499

Generation: 8 - 38: 188024

New Results

Colony size: 20
Iteration: 50
ph_value: 0.5
ev_parameter: 0.2
delta: 0.1
alpha: 1
beta: 1
pop _size: 100

Roulette Wheel

best result: 49205

Problems and Discussion

- Ant algorithm is very slow
- we tossed the GA
- heuristics cannot be easily substituted
  - before: everything was random and we hoped to get better results with the help of the GA
  - now: choose heuristics in the initialization and found decent solutions

Results

ACO parameters:
ant number = 5
iterations = 5
delta = 0.1
ph_value = 0.5
alpha = 0.5
beta = 0.2

GE parameters:
roulette wheel
uniform crossover
mutator = random resetting
population size = 50
crossover rate = 1
mutation rate = 0.4
ev_parameter = 0.2

Development over Generations:

Generation 1-x: 277346

Results

ACO parameters:
ant number = 10
ev_parameter = 0.2
iterations = 10
delta = 0.1
ph_value = 0.5
alpha = 1
beta = 0.5

GE parameters:
roulette wheel
uniform crossover
mutator = random resetting
population size = 25
crossover rate = 1
mutation rate = 0.6

Development over Generations:

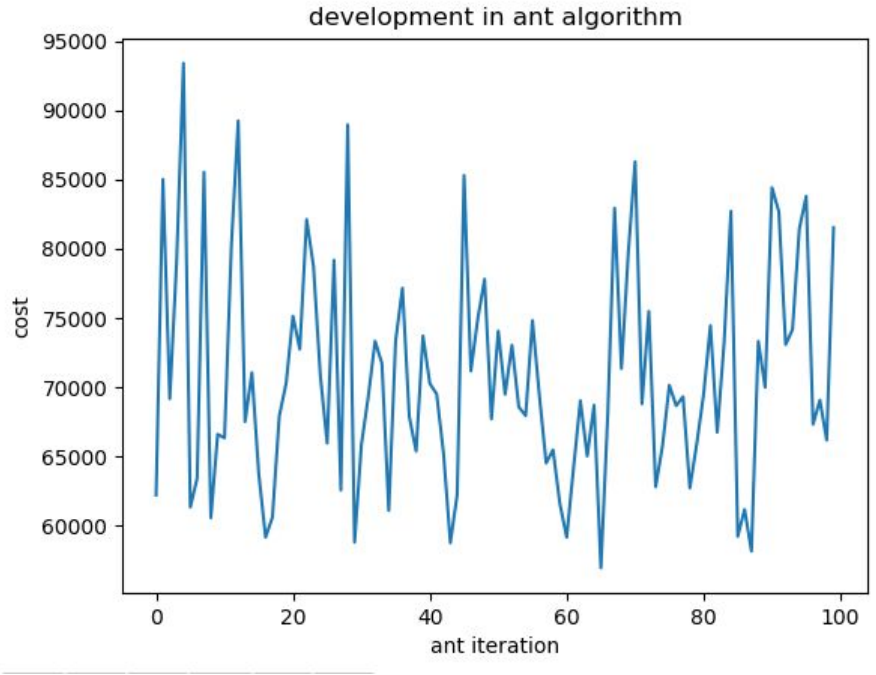
Generation 1-8..: 174064

173288

Results new

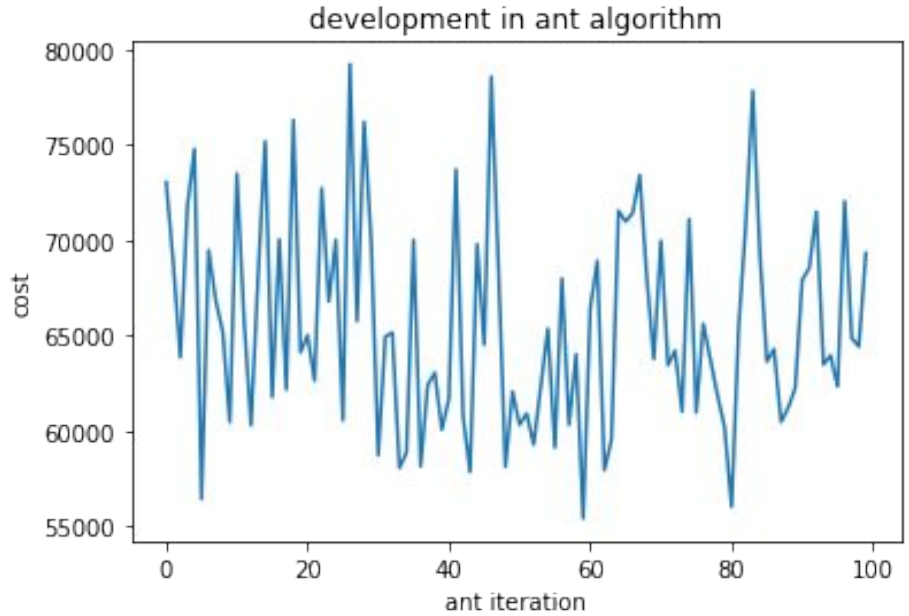
Parameters

population size: 100
number of ants: 20
car choosing mode: random
alpha: 1
beta: 0.5
delta: 0.1
path-choosing mode: roulette wheel
minimal cost (best solution): 56972



Results New

Parameters:
number of ants:: 20
Iteration: 50
car choosing mode: random
delta: 0.1
alpha: 1
beta: 1
population size: 100
path choosing mode: roulette wheel
intensification: all-ants
minimal cost (best solution): 55419

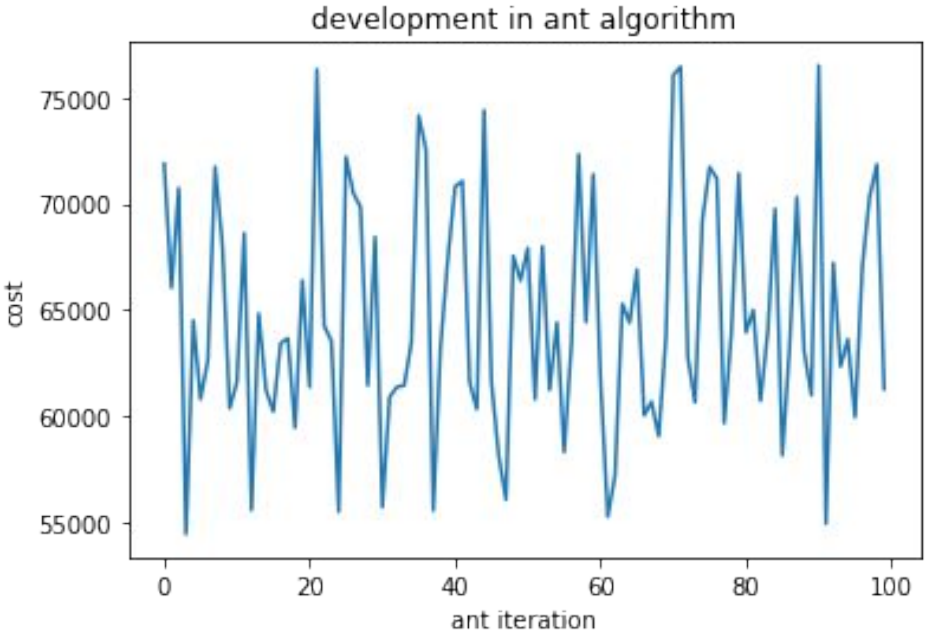


Results New

Parameters:
number of ants: 20
Iteration: 50
car choosing mode: random
delta: 0.1
alpha: 1
beta: 1
population size: 100

path choosing mode: max. Probability

minimal cost (best solution): 54480



Results New

Parameters:
number of ants: 100
Iteration: 50
car choosing mode: high-middle
delta: 0.1
alpha: 1
beta: 1
population size: 100

path choosing mode: max. Probability

minimal cost (best solution): 51495
Results