Erkan Çetinyamaç

UYGULAMALI KARAR MODELLERİ

Vehicle Routing Probleminin Tabu Search ile Çözümünün Uygulaması

Formulasyon

Çözüm, droneların atanan teslimatların teslimine yönelik vektör grafiğini çizdirmek suretiyle olacaktır.

Amaç Fonksiyonu

$$F = \min\{\sum_{i=1}^{M}\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij}x_{ij}\}$$

burada:

- M : path sayısı.
- N : alıcı sayısı.
- c_{ij} : i'den j'ye giderkenki cost.
- x_{ij} : a dronunun i'den j'ye yönlendirilip yönlendirilmediğini belirleyen karar değişkeni.
- $x_{ij} = \begin{cases} 1 & \text{drone 'i'den j'ye giderken.} \\ 0 & \text{diğer durumlar.} \end{cases}$

Kısıtlar

Eğer $x_{ij} = 1 \Rightarrow u_i + q_{ij} = u_j$</br> bir drone tarafından ziyaret edilen müşteriye ait dağıtım ve toplamanın aynı araç tarafından yapılması gerektiğini göstermektedir

$$q_{ik} \leq u_i \leq Q \quad \forall_i \in 1,2,...,N$$

q_{ik} – i. müşteriye by e. drone tarafından teslim edilen talep miktarı.

Q – drone kapasitesi.

Veri Yapıları

> M: Mesafe (Distance) Matrisi

$$M_{ij} = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i,1} & d_{i,2} & \cdots & d_{i,j} \end{pmatrix}$$

burada:

- i, j - Alıcıların İndex Tanımları
- $d_{i,j}$ - Alıcılar Arasındaki Mesafe i, j
- Öklid mesafesi $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

> her bir alıcının koordinat listesini

$$L_i = a_1, a_2, ... a_i$$

temsil eder.

burada:

- (x_i, y_i) a_i - i. alıcının koordinatını

temsil eder.

In [1]:
```
from tabu_search import TabuSearch
from visualization import WithVisualization
from matplotlib import pyplot as plt
import numpy as np
```

Algoritma Parametrelerinin Testi

In [2]:
```
# Sabit değişken ve fonksiyon parametre tanımları.
MAX_COST = 99999
MIN_TABU_SIZE = 50
MAX_TABU_SIZE = 52
TABU_STEP = 1
NUM_OF_TESTS = 3
N_ITERS = 5000
FILE_NAME = "test_data.txt"
NUM_DRONES = 8
DRONE_CAPACITY = 4
NUM_CLIENTS = 30

# Hafıza (Memory) Değişkenlerinin tanımları. Arama boyunca ortaya çıkan durumlar hafıza değişkenlerinde
 tutulacaktır.
best_of_all = MAX_COST
size = NUM_TABU_SIZE
costs_history = []
best_costs_history = []
fitness_history = []
tabu_size_average_costs = {}
best_model_result = None
tabu_size = None



while size <= MAX_TABU_SIZE:
    for _ in range(NUM_OF_TESTS):
        ts = TabuSearch(NUM_DRONES, DRONE_CAPACITY, NUM_CLIENTS, FILE_NAME)
        ts.search(tabu_size=size, n_iters=N_ITERS)
        best_cost = ts._fitness(ts.best_solution)
        print(f'Tabu boyutu: {size}, En iyi cost: {best_cost}')
        fitness_history.append(ts.best_cost)
        best_costs_history.append(ts.best_costs)
        costs_history.append(ts.costs)
        if best_cost < best_of_all:
            best_of_all = best_cost
            best_tabu_size = size
            best_model_result = ts
    hist_arr = np.array(fitness_history)
    tabu_size_average_costs[size] = hist_arr.mean()
    fitness_history = []
    print("=======================")
    print(f'Ortalama: {tabu_size_average_costs[size]}')
    print(f'En iyi: {hist_arr.min()}\n')
    size += TABU_STEP
print(f"En iyi COST: {best_of_all} | Tabu boyutu: {best_tabu_size} | İterasyon sayısı: {N_ITERS}")
```

```
Elapsed time 6.526239709243774 ms
Tabu boyutu: 50, En iyi cost: 535.4680340793713
Elapsed time 6.197655440734863 ms
Tabu boyutu: 50, En iyi cost: 535.4680340793713
Elapsed time 6.649641990661621 ms
Tabu boyutu: 50, En iyi cost: 545.2341267114126

Ortalama: 538.7233982901183
En iyi: 535.4680340793713

Elapsed time 7.024263364314 ms
Tabu boyutu: 51, En iyi cost: 535.4680340793713
Elapsed time 6.950408697128296 ms
Tabu boyutu: 51, En iyi cost: 540.6311889318011
Elapsed time 7.028812408447266 ms
Tabu boyutu: 51, En iyi cost: 535.4680340793713

Ortalama: 537.1890850698479
En iyi: 535.4680340793713

Elapsed time 6.959020853042625 s
Tabu boyutu: 52, En iyi cost: 536.7381320292432
Elapsed time 6.580034971237183 s
Tabu boyutu: 52, En iyi cost: 540.6311889318011
Elapsed time 6.692744660377502 s
Tabu boyutu: 52, En iyi cost: 535.4680340793713

Ortalama: 537.6124516801384
En iyi: 535.4680340793713

En iyi COST: 535.4680340793713 | Tabu boyutu: 50 | İterasyon sayısı: 5000
```

In [3]:
```
%matplotlib inline
```

Test Sonuçlarının Plotları

In [4]:
```
# Her bir test için cost fonksiyonunun hafızasının plotu.


fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(13, 13))
for i, (c, b) in enumerate(zip(costs_history[0:5], best_costs_history[0:5])):
    ax1.plot(c[:5:], label=f'Test {i} - En optimum (min(b) :.2f}')
    ax2.plot(b[:5:], label=f'Test {i}')
ax2.legend(), ax1.legend()
ax1.grid(), ax2.grid()
ax1.set_title("İterasyon sayısına göre cost fonksiyonunun değişimi", fontdict={'fontsize': 14})
ax2.set_title("Tüm iterasyonlar sayısında en iyi cost fonksiyonu", fontdict={'fontsize': 14})
plt.show()
```


En İyi (Optimum) Sonucun Plotu

In [5]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(best_model_result.costs[0:], label="İterasyonlar içerisindeki costların değişimi")
ax.plot(best_model_result.best_costs[0:], label="İterasyonlar içerisindeki en iyi (optimum) cost")
ax.set_title(f"Cost fonksiyonun  en iyi (optimum) sonucu( best_of_all.2f}")
ax.legend()
plt.xlabel('İterasyon')
plt.ylabel('Cost')
plt.grid()
plt.show()
```


En İyi (Optimum) Sonucun Görselleştirilmesi

In [6]:
```
vis = WithVisualization(best_model_result)
vis.plot_figure(sizes=(10, 10))
vis.plot_solution()
```


In [7]:
```
%matplotlib qt
```

En İyi (Optimum) Sonucun Simülasyon Plotu

In [ ]:
```
logs = open("logs.txt", "w")
logs.close()

vis = WithVisualization(best_model_result)
vis.visualize_solution()
```

```
Time: 1 min  (Drone: 1 ) Packages: 4) - In base
Time: 1 min  (Drone: 2 ) Packages: 4) - In base
Time: 1 min  (Drone: 3 ) Packages: 4) - In base
Time: 1 min  (Drone: 4 ) Packages: 4) - In base
Time: 1 min  (Drone: 5 ) Packages: 4) - In base
Time: 1 min  (Drone: 6 ) Packages: 4) - In base
Time: 1 min  (Drone: 7 ) Packages: 4) - In base
Time: 1 min  (Drone: 8 ) Packages: 4) - In base
Time: 11 min  (Drone: 6 ) Packages: 3) - Package delivered to client with id 19
Time: 12 min  (Drone: 8 ) Packages: 3) - Package delivered to client with id 16
Time: 14 min  (Drone: 6 ) Packages: 2) - Package delivered to client with id 22
Time: 15 min  (Drone: 7 ) Packages: 3) - Package delivered to client with id 6
Time: 16 min  (Drone: 3 ) Packages: 3) - Package delivered to client with id 3
Time: 16 min  (Drone: 8 ) Packages: 2) - Package delivered to client with id 27
Time: 21 min  (Drone: 3 ) Packages: 2) - Package delivered to client with id 7
Time: 22 min  (Drone: 7 ) Packages: 2) - Package delivered to client with id 18
Time: 23 min  (Drone: 1 ) Packages: 3) - Package delivered to client with id 2
Time: 25 min  (Drone: 6 ) Packages: 1) - Package delivered to client with id 12
Time: 26 min  (Drone: 8 ) Packages: 4) - In base
Time: 28 min  (Drone: 3 ) Packages: 1) - Package delivered to client with id 24
Time: 28 min  (Drone: 4 ) Packages: 3) - Package delivered to client with id 29
Time: 28 min  (Drone: 5 ) Packages: 3) - Package delivered to client with id 25
Time: 32 min  (Drone: 3 ) Packages: 0) - Package delivered to client with id 8
Time: 34 min  (Drone: 2 ) Packages: 3) - Package delivered to client with id 1
Time: 37 min  (Drone: 2 ) Packages: 2) - Package delivered to client with id 10
Time: 39 min  (Drone: 4 ) Packages: 2) - Package delivered to client with id 30
Time: 40 min  (Drone: 1 ) Packages: 2) - Package delivered to client with id 9
Time: 40 min  (Drone: 7 ) Packages: 1) - Package delivered to client with id 23
Time: 41 min  (Drone: 6 ) Packages: 0) - Package delivered to client with id 5
Time: 44 min  (Drone: 5 ) Packages: 2) - Package delivered to client with id 20
Time: 48 min  (Drone: 3 ) Packages: 4) - In base
Time: 48 min  (Drone: 7 ) Packages: 0) - Package delivered to client with id 13
Time: 49 min  (Drone: 2 ) Packages: 1) - Package delivered to client with id 26
Time: 49 min  (Drone: 4 ) Packages: 1) - Package delivered to client with id 14
Time: 50 min  (Drone: 1 ) Packages: 1) - Package delivered to client with id 11
Time: 53 min  (Drone: 2 ) Packages: 0) - Package delivered to client with id 17
```

In [ ]:
```

```