Zoekprobleem autosharing: Verslag

Pieter Dilien, Lucas Van Laer

29 maart 2023

1 Oplossingsvoorstelling

De oplossing wordt voorgesteld door een object dat 3 attributen heeft. De eerste is een req_to_car array. Hierbij stelt elke index een reservatie voor en elke waarde een wagen. De tweede is een car_to_zone array. Hierbij stelt elke index een wagen voor en elke waarde een zone. De laatste is een int die de cost voorstelt.

Hieronder de oplossingsvoorstelling zowel in Python als in Rust.

```python
#Python
class SolutionModel:
    req_to_car: npt.NDArray[np.int16]   # Give req as index get car
    car_to_zone: npt.NDArray[np.int16]  # Give car as index get zone
    cost: int

//Rust
pub struct SolutionModel<'a> {
    pub req_to_car: Vec<i64>,
    pub car_to_zone: Vec<i64>,
    pub cost: i64
}
```

Deze voorstelling is gebaseerd op de output-file dat gegenereerd moet worden. Daardoor is het wegschrijven naar deze file dus erg gemakkelijk.

2 Datavoorstelling

3 Initiele Oplossing

Om de initiele oplossing te berekenen gaan we sequentieel over de lijst van reservaties. Bij elke reservatie gaan we de verschillende wagens af. Eerst wordt er nagegaan of de wagen reeds aan een zone gelinkt is, indien dit nog niet het geval is, plaatsen we de wagen in de zone van de reservatie.

Indien de wagen al aan een zone gelinkt is checken we of deze in de (naburige) zone van de reservatie ligt. Indien dit het geval is wordt er gekeken of deze wagen dan ook beschikbaar is, zo ja linken we deze wagen aan de reservatie.

Als er over alle wagens (die mogelijk zijn voor de specifieke reservatie) gegaan is en er geen wagen beschikbaar is, wordt er naar de volgende reservatie gegaan en blijft de reservatie ontoegewezen.

4 Zoekomgeving

4.1 Kleine Operator

4.2 Grote Operator

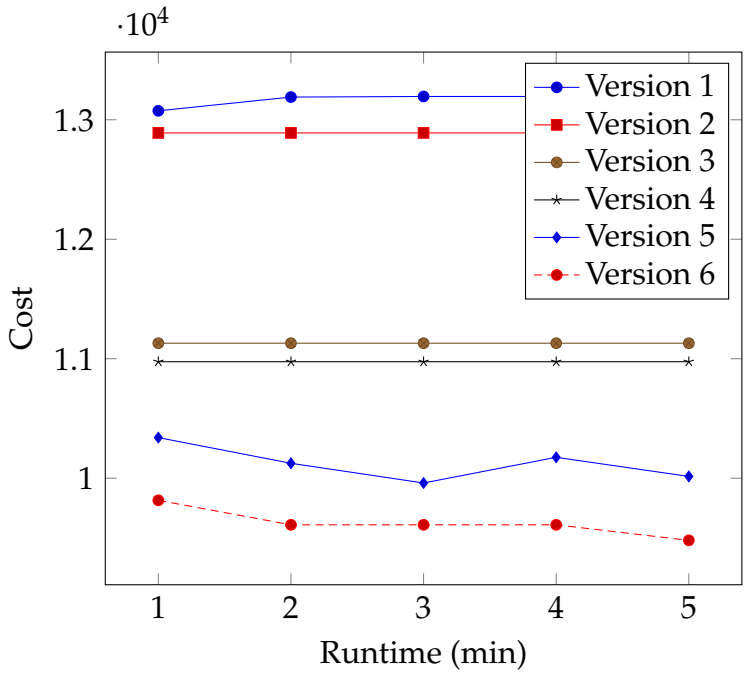
5 Metaheuristiek

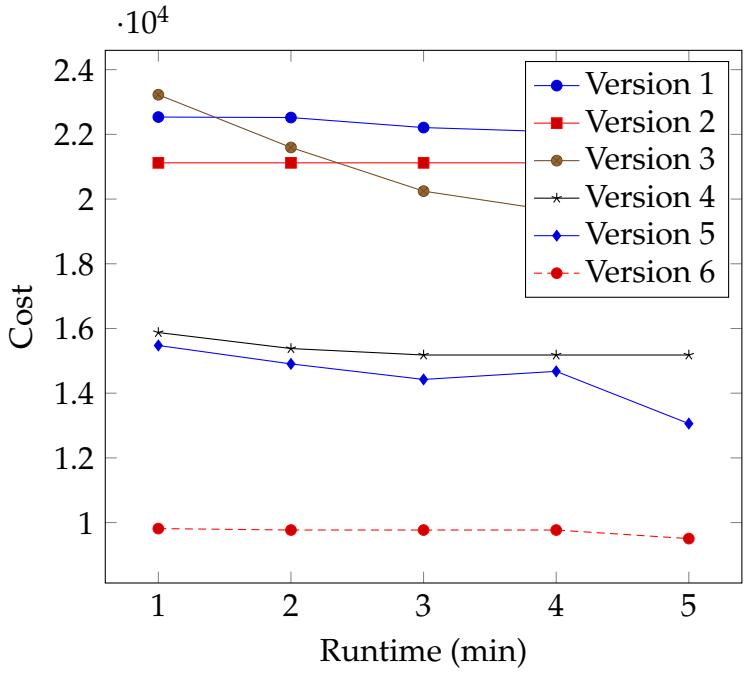
5.1 Simulated Annealing

5.2 Thresholding

6 Resultaten

7 Bijlagen




·10^4
2.4
2.2
2
1.8
1.6
1.4
1.2
1
Cost
1
2
3
4
5
Runtime (min)
Version 1
Version 2
Version 3
Version 4
Version 5
Version 6