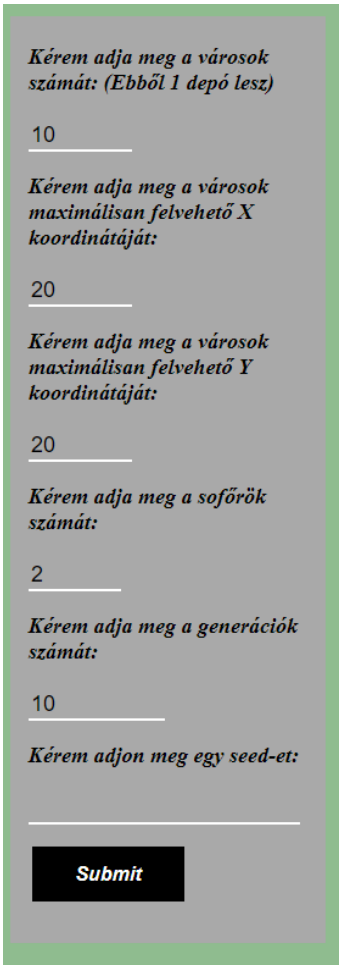
Korom Tamás

N7D1L5

Mesterséges intelligencia beadandó
Dokumentáció

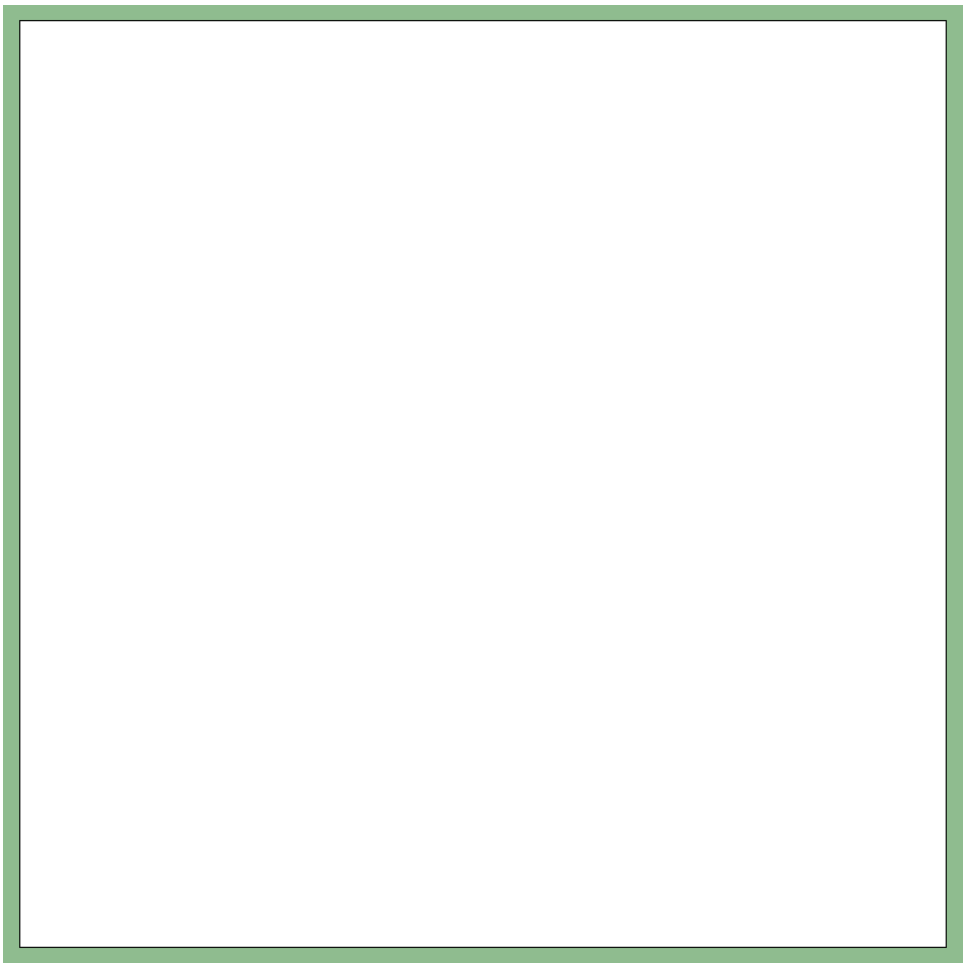
Bevezetés

A beadandó feladat problémája az VRP volt amit genetikus algoritmus segítségével kellett megoldani. Az általam választott programozási nyelv a Javascript volt. Megjelenítéshez pedig egy egy oldal szolgál ami egy React oldal.



Ahogy a mellékelt képen is látszik megtudjuk adni az adatokat amit elküld egy webszervernek és az majd válaszolni fog majd a legjobb útvonalal majd.

A webszer által elkészített legjobb útvonalat majd egy Canvas-ra fogja majd rajzolni.



Megoldás menete:

Az elején a program a megadott kezdő paraméterekkel legenerálja a városok koordinátáit és azt egy tömbe fog majd letárolodni. A generálások seed-hez vannak kötve ezért bármikor reprodukálható:

```
const seed = require("random-seed");

function generator(size, citiesDistance_x, citiesDistance_y, seedData){

    var array = [{}]

    const seedInit = seed.create()

    seedInit.seed(seedData)
    for(i = 0; i < size; i++) {
        array[i] = {
            x: seedInit(citiesDistance_x),
            y: seedInit(citiesDistance_y),
            counter: 0,
        }
    }

    return array;
}
      You, 3 héttel ezelőtt • init …
module.exports = {
    generator
}
```

A meglévő városokkal kiszámítok egy lehetséges megoldást a mohó algoritmus segítségvel.

```
for(k = 0; k < drivers; k++){

    const route = {driverRoute: [0], coords: [points[0]], distance: 0, probability: 0, rand: 0, cr
    let base = 0
    let sumDistance = 0

    for(i = 0 ; freeCount != 0 && i < node; i++){
        let min_x = points[base].x
        let min_y = points[base].y

        let distance = Infinity
        let coord = null

        for(j = 1; j < points.length; j++){
            if(base != j && points[j].counter == 0 && (Math.abs(min_x - points[j].x) + Math.abs(mi
                distance = Math.abs(min_x - points[j].x) + Math.abs(min_y - points[j].y)
                base = j
                coord = points[j]
            }
        }
        sumDistance += distance

        route.driverRoute.push(base)
        route.coords.push(coord)

        points[base].counter++
        freeCount -= 1

    }
    route.driverRoute.push(0)
    route.coords.push(points[0])
    sumDistance += Math.abs(points[0].x - points[base].x) + Math.abs(points[0].y - points[base].y)

    allSumDistance += sumDistance

    route.distance = sumDistance
    routes.push(route)
}

return geneticAlgorithm(routes, allSumDistance, points, generationCount)
```

Ahogy az ábrán is látható a manhatten távolságot alapúl véve számoltam ki a távolságokat és mindig a legrövidebbet választottam.

```
for(i = 0 ; freeCount != 0 && i < node; i++){
    let min_x = points[base].x
    let min_y = points[base].y

    let distance = Infinity
    let coord = null

    for(j = 1; j < points.length; j++){
        if(base != j && points[j].counter == 0 && (Math.abs(min_x - points[j].x) + Math.abs(mi
            distance = Math.abs(min_x - points[j].x) + Math.abs(min_y - points[j].y)
            base = j
            coord = points[j]
        }
    }
    sumDistance += distance

    route.driverRoute.push(base)
    route.coords.push(coord)

    points[base].counter++
    freeCount -= 1

}
```

A mikor meghívom erre a megoldásra a genetikus algoritmust kiszámolneki egy valószínűséget ez a számolás pedig a sofőr útvonal hossza osztva a teljes útvonal hosszal. Eztuán hozzá rendelek még egy random értéket.

```
const {probabilityCalc, rand} = require("./function")

const fitnesAndRand = ((population, sumDistance) => {
    for(let i = 0; i < population.length; i++){
        population[i].probability = probabilityCalc(population[i].distance, sumDistance)
        population[i].rand = rand()
    }
})

module.exports = {
    fitnesAndRand       You, 14 órával ezelőtt • függvénybe szét válogatás
}
```

Amikor ezek megtörténtek minden egyednek választok egy párt ügyelve arra hogy saját magával ne kersztezze majd.

```
const selection = ((population) => {
    for(let i = 0; i < population.length; i++){
        let selection = 0
        for(j = 0; j < population.length; j++){
            selection += population[j].probability

            if(population[i].rand <= selection && population[i].parent != true && population[j].parent
                population[i].crossOver = j
                population[i].parent = true
                population[j].crossOver = i
                population[j].parent = true
            }       You, 14 órával ezelőtt • függvénybe szét válogatás …
        }
    }
})

module.exports = {
    selection
}
```

Revorveles valószínűséggel számoltam.

Miután megtörténtek a kiválasztások a megfelelő egyedeket keresztezem egy mással.

```
const crossing = ((population) => {

    for(let i = 0; i < population.length; i++){
        if(population[i].crossOver > i){
            let index = population[i].crossOver
            let firstElement = Math.floor(Math.random() * (population[i].driverRoute.length - 2) + 1)
            let secondElement = Math.floor(Math.random() * (population[index].driverRoute.length - 2)
            let range = 3

            for(let k = 0; k < range; k++){
                if(firstElement + k > population[i].driverRoute.length - 2){
                    firstElement = 1
                }
                if(secondElement + k > population[index].driverRoute.length - 2){
                    secondElement = 1
                }

                let temp = population[i].driverRoute[firstElement + k]
                population[i].driverRoute[firstElement + k] = population[index].driverRoute[secondElem
                population[index].driverRoute[secondElement + k] = temp

                temp = population[i].coords[firstElement + k]
                population[i].coords[firstElement + k] = population[index].coords[secondElement + k]
                population[index].coords[secondElement + k] = temp
            }
        }
    }
})

module.exports = {
    crossing
}       You, 14 órával ezelőtt • függvénybe szét válogatás …
```

A keresztezések után minden egyeden egy mutációt hajtunk végre.

```
const { distanceCalc } = require("./function")

const mutation = ((population, points) => {
    for(let i = 0; i < population.length; i++){
    let firstElement = Math.floor(Math.random() * (population[i].driverRoute.length - 2) + 1)
    let secondElement = 0

    do{
        secondElement = Math.floor(Math.random() * (population[i].driverRoute.length - 2) + 1)
    }
    while(firstElement == secondElement)

    let temp = population[i].driverRoute[firstElement]      You, 14 órával ezelőtt • függvénybe szét
    population[i].driverRoute[firstElement] = population[i].driverRoute[secondElement]
    population[i].driverRoute[secondElement] = temp

    temp = population[i].coords[firstElement]
    population[i].coords[firstElement] = population[i].coords[secondElement]
    population[i].coords[secondElement] = temp

    population[i].distance = distanceCalc(population[i].driverRoute, points)
    }
})

module.exports = {
    mutation,
}
```

Végezetül a function.js-ben találhatóak meg ilyen alap függvények amit felhasználtam és hogy később is használhatóak legyenek.

```
function distanceCalc (driverRoute, array) {
    let distance = 0
    for(let i = 1; i < driverRoute.length; i++){
        distance += Math.abs(array[driverRoute[i - 1]].x - array[driverRoute[i]].x) + Math.abs(array[d
    }
    return distance
}

function probabilityCalc (distance, sumDistance) {
    return Number((distance / sumDistance).toFixed(4))
}

function rand() {
    return Number((Math.random()).toFixed(4))
}

function allDrivedDistance(population) {
    let allDistance = 0
    for(let i = 0; i < population.length; i++) {
        allDistance += population[i].distance
    }
    return allDistance
}       You, 5 nappal ezelőtt • Calc func …


//legjobb indexű generéció kiválasztása
function min(generations){
    let index = 0
    let bestRoutesDistance = allDrivedDistance(generations[index].generation)
    for(let i = 0; i < generations.length; i++){
        if(bestRoutesDistance > allDrivedDistance(generations[i].generation)) {
            index = i
            bestRoutesDistance = allDrivedDistance(generations[i].generation)
        }
    }
    return index
}


module.exports = {
    distanceCalc,
    probabilityCalc,
    rand,
    allDrivedDistance,
    min
```

A szerverek elinditásához először lépjünk be a ./MesInt/gui mappába és adjuk ki az **npm start** parancsot ezzel a weboldal fog elindulni majd a localhost:3000-en majd,

A backend elinditásához is hasonló az eljárás csak annyi hogy ezt a ./MesInt/Vehicle_Routing_Problem mappán tudjuk megtenni ugyan úgy az **npm start** parancsal itt nodemon fut szóval ha bele írunk a kódba és rá mentünk a szerver automatikusan újra indul a változtatásokkal.