Mesterséges Intelligenciák féléves feladat

WH85ZH - Nehéz

2022/23 I. félév

A feladat implementációját – azaz a forráskódot, és a lezáró dokumentumot az `aitflew@uni-miskolc.hu` e-mail címre kell elküldeni, majd azt a 13. és 14. héten a gyakorlatokon kell megvédeni. A kész feladat leadásához a GitHub javasolt, de nem kötelező. Az e-mail tárgya és a küldő jól beazonosítható legyen, ez vonatkozik a GitHub felhasználóra is.

Tetszőleges nyelv, keretrendszer, technológia választható. Az egyetlen megkötés, hogy nem lehet olyan könyvtárat használni, amely tartalmazza a feladat modelljét és/vagy a megoldó algoritmusokat.

A plusz feladatok elvégzésével magasabb érdemjegyet lehet elérni.

Probléma: Vehicle Routing Problem with Capacities (CVRP)

A CVRP probléma a Vehicle Routing Problem-re épül azzal a megkötéssel, hogy a futároknak kapacitásuk van, a városoknak pedig kapacitásigényük.

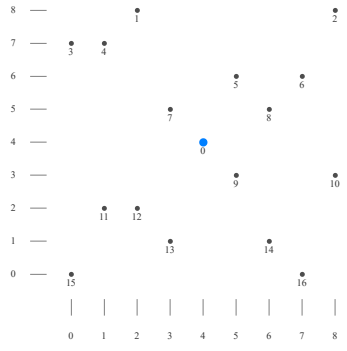
A VRP probléma

A Vehicle Routing Problem (VRP), a Traveling Salesman Problem (TSP) általánosítása. Az ügynök helyett több „futár" megy egy bázisból a városokba. Fontos, hogy a városok mindegyikében egy, és csakis egy futárnak kell járnia. A bázis az egyetlen pont, amelyet több futár érinthet. Mivel a bázis ez esetben különbözik a többi várostól, ezért a modellben külön kell kezelni.

A kereső algoritmusban használt n-opt (a gyakorlatokon a 2-opt került bemutatásra) operátor kétféle kell, hogy legyen. Mivel a futárok számával egyenlő útvonal van, ezért az egyes útvonalakon belül és az útvonalak közt is cserélnünk kell.

Egy lehetséges Vehicle Routing probléma, ahol a depó a 0 számú csomópont és a járművek/futárok száma 4. Ennek egy leírása Python nyelven:

```
[(456, 320), # location 0 - the depot
(228, 0),    # location 1
(912, 0),    # location 2
(0, 80),     # location 3
(114, 80),   # location 4
(570, 160),  # location 5
(798, 160),  # location 6
(342, 240),  # location 7
(684, 240),  # location 8
(570, 400),  # location 9
```



1. ábra: A probléma grafikusan ábrázolva

```
(912, 400),  # location 10
(114, 480),  # location 11
(228, 480),  # location 12
(342, 560),  # location 13
(684, 560),  # location 14
(0, 640),    # location 15
(798, 640)]  # location 16
```

Ekkor a városok közti távokat Manhattan távolságként adjuk meg. (x_1, y_1) és (x_2, y_2) Manhattan távolsága: $|x_1 - x_2| + |y_1 - y_2|$.

A probléma egyik lehetséges megoldása:

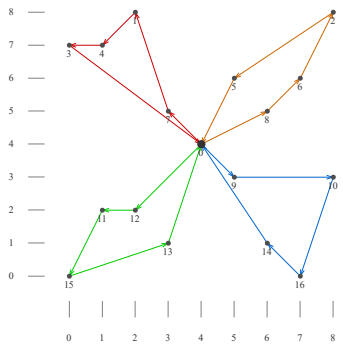
```
Route for vehicle 0:
 0 -> 8 -> 6 -> 2 -> 5 -> 0
Distance of route: 1552m

Route for vehicle 1:
 0 -> 7 -> 1 -> 4 -> 3 -> 0
Distance of route: 1552m

Route for vehicle 2:
 0 -> 9 -> 10 -> 16 -> 14 -> 0
Distance of route: 1552m

Route for vehicle 3:
 0 -> 12 -> 11 -> 15 -> 13 -> 0
Distance of route: 1552m

Total distance: 6208m
```



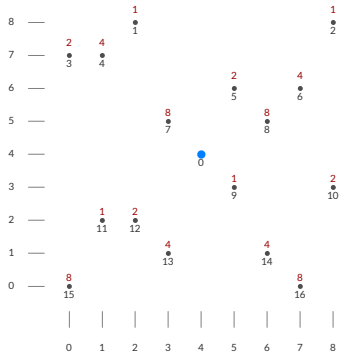
2. ábra: A megoldás grafikusan ábrázolva

Megkötések

A VRP problémában megadott városokhoz keresletet is rendelunk, valamint a futárokhoz kapacitást.

Ez Python nyelven az alábbi módon nézhet ki:

```
demands =
    [0, 1, 1, 2, 4, 2, 4, 8, 8, 1, 2, 1, 2, 4, 4, 8, 8]
vehicle_capacities = [15, 15, 15, 15]
```

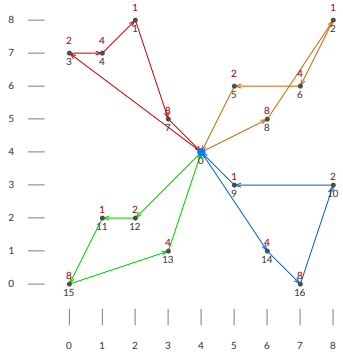


3. ábra: A probléma grafikusan ábrázolva

A probléma egyik lehetséges megoldása:

```
Route for vehicle 0:
 0 Load(0) -> 4 Load(0) -> 3 Load(4)
 -> 1 Load(6) -> 7 Load(7) -> 0 Load(15)
Distance of route: 1552m
Load of the route: 15

Route for vehicle 1:
 0 Load(0) -> 14 Load(0) -> 16 Load(4)
 -> 10 Load(12) -> 9 Load(7) -> 0 Load(15)
```



4. ábra: A megoldás grafikusan ábrázolva

```
Distance of route: 1552m
Load of the route: 15

Route for vehicle 2:
 0 Load(0) -> 12 Load(0) -> 11 Load(2)
 -> 15 Load(3) -> 13 Load(11) -> 0 Load(15)
Distance of route: 1552m
Load of the route: 15

Route for vehicle 3:
 0 Load(0) -> 8 Load(0) -> 2 Load(8)
 -> 6 Load(9) -> 5 Load(13) -> 0 Load(15)
Distance of route: 1552m
Load of the route: 15

Total distance: 6208m
Total Load of all routes: 60
```

Ezeket a feladatokat inicializálja valamilyen véletlen szám generátorral! Figyeljük meg, hogy nem minden megoldás lesz jó, lehetsége, hogy túllépjük a futárok kapacitását. A feladat generálásánál is oda kell figyelni, hogy az összes kereslet ne legyen több az össz kapacitásnál. Legyen különböző méretűek:

- a városok száma legyen 10, 20, 50, 100, 200, 500,
- a futárok száma legyen 1 (TSP), 2, 4, 5, nagyobb feladatok esetén (ahol a városok száma legalább 50): 10, 20.

A legenerált feladatok legyenek perzisztensen tárolva (vagy seedhez kötéssel procedurálisan generálva).

Algoritmus: Szimulált Hűtés

A feladatot a Szimulált Hűtés algoritmussal kell megoldani. Ez az algoritmus az egyszerű szomszédsági keresésre épül, azzal a különbséggel, hogy a keresés során valamilyen valószínűségtől függ, hogy elfogadunk-e rosszabb eredményt. Ennek a valószínűségnek a kiszámítása függ az elért eredménytől és a hőmérséklettől:

$$P_t = e^{-\frac{f^* - f(s_t)}{k * T_t}},$$

ahol t az eltelt idő (iteráció száma), f^* az eddigi legjobb eredmény, $f(s_t)$ a jelenlegi eredmény, k az ún. Boltzmann konstans, T_t pedig a t-edik időben a hőmérséklet.

A hőmérsékletet többféleképpen változtathatjuk. Ez hűtési stratégiának hívjuk. Lehet monoton és nem monoton is a hőmérséklet.

Plusz feladatok

Válasszon több hűtési stratégiát és vesse össze őket az adott problémán!

Ezeket a stratégiákat használja nem monoton hűtésnél, mely a jelenlegi eredménytől függ:

$$T(t) = \mu T_t = \left(1 + \frac{f(s_i) - f^*}{f(s_i)} \cdot T_t\right)$$

Az algoritmus végét egy ún. leállási feltétel vizsgálatával érjük el. Ez lehet egy adott iterációszám elérése, egy előre meghatározott hőmérséklet elérése, valamennyi iteráci eltelte úgy, hogy nem javult az eredmény, vagy ezeknek valamilyen kombinációja. Vessen össze ezekből legalább kettőt, vagy valamilyen kombinációikat. Hogyan hat ez az eredményre és a futási időre?

Készítsen egy alkalmazást, ahol a feladat megadásával elkészíti autómatikusan a megoldást és ezt valahogy megjeleníti (térkép, Gantt diagram).

A feladat lezárása

A féléves feladat lezárásaként egy dokumentumot kell elkészíteni, melyben szerepelnek a megoldás lépései, a választott nyelv, technológia, könyvtárak, keretrendszerek. Esetlegesen, a külön irodalomkutatás eredményeit is tartalmazza.

Minden plusz munka javítja a kapható érdemjegyet. A könnyű feladat csak kivételes esetben érhet el hármasnál jobb jegyet.