Mesterséges Intelligenciák féléves feladat

WYQ5JK - Könnyű

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.

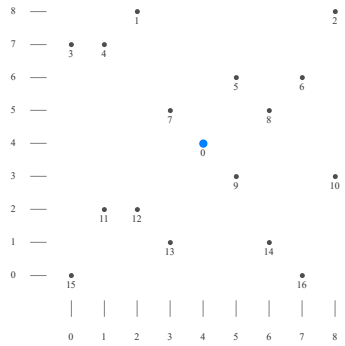
Probléma: Vehicle Routing Problem

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
(912, 400),  # location 10
(114, 480),  # location 11
(228, 480),  # location 12
(342, 560),  # location 13
(684, 560),  # location 14
```



1. ábra: A probléma grafikusan ábrázolva

```
(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: $|x_1 - x_2| + |y_1 - y_2|$.

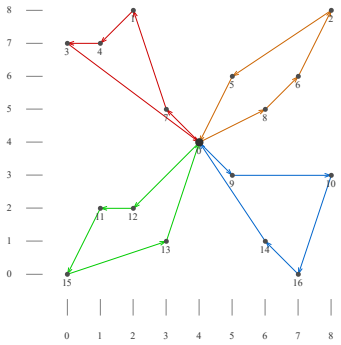
A probléma egyik lehetséges megoldása:

```
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
```

Ezeket a feladatokat inicializálja valamilyen véletlen szám generátorral! 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).



2. ábra: A megoldás grafikusan ábrázolva

Algoritmus: Tabu keresés

A feladatot a Tabu kereséssel kell megoldani. Ez az algoritmus az egyszerű szomszédsági keresésre épül, azzal a különbséggel, hogy a keresés során bevezetünk egy tabu listát, melyben tároljuk, hogy mik azok a bázisok, ahol már jártunk. Ezt determinisztikus elfogadási kritériumnak nevezzük. Ez a lista változásokat, vagy eredményeket is tarthat nyilván.

Ehhez a tabulistához általában rendelűnk egy maximális elemszámot. Ha ez az elemszám nagyon nagy, akkor a keresés túlságosan korlátozottá válik, ha túl kicsi, akkor ismétlődő keresési útvonal alakulhat ki.

Persze ezt nem csak mérettel lehet korlátozni. Lehetséges, hogy az egyes tiltott megoldásokhoz egy látogatási számot rendelünk. Ha a látogatások száma elér egy korlátot, akkor kivesszük a tabu listából.

Az elemek reprezentációja a listában lehet sokféle, mely a feladattól függ. Lehet egy kiindulási pont és a hozzá tartozó tiltott átalakítás, vagy átalakítások. Lehetnek a tiltott eredmények, vagy a tiltott eredmények valamilyen kódolt formája (akár hash is).

Plusz feladat

Vizsgálja meg, hogy tabu lista méretének változtatásával hogyan változik az eredmény, a futási idő, és a memóriaigény!

Vizsgálja meg, hogy mi történik az egyes kikerülési kritériumoknál! Ha fix a tabulista, ha csak azok vannak a tabulistában, amikre már sokszor sor került, ha az egyes elemek csak t ideig vannak a listában, ha n látogatás után kikerülnek az elemek.

A tabu listából a kikerülés kritériumainak paraméterei változhatnak a keresés során. Próbáljon ki különböző stratégiákat arra, hogy ezek hogyan változzanak a keresés során. Ehhez érdemes tanulmányozni a szimulált hűtés hűtési stratégiáit.

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.