Mesterséges Intelligenciák féléves feladat

D3U3EE - 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: Flow-shop ütemezés

A flow-shop feladat egy ún. egyutas, többoperációs gyártásütemezési feladat. A gyártásütemezési feladatok formális leírására az alfa, béta, gamma jelölést alkalmazzuk ($\alpha|\beta|\gamma$). E szerint a rendszer szerint az egyszerű flow-shop feladat leírása:

$$F||C_{max}$$

Jelentése:

Az erőforrások az ütemezési időszakban folyamatosan rendelkezésre állnak. Az erőforrások egyszerre csak egy munkán dolgoznak. A munkák legkorábbi indítási időpontja nulla (bármikor indíthatóak). Minden egyes munkához adott m számú operáció tartozik, melyeknek pontosan ismert a végrehajtási ideje: $p_{i,j}, i \in 1, 2, \ldots, n, j \in 1, 2, \ldots, m$. Az operációk végrehajtási sorrendje kötött és minden munka esetében azonos. Az operációk végrehajtása nem szakítható meg. A gépek között a munkák várakozhatnak, a műveletkőzi tárolók mérete nem korlátos. Az ütemezés célja az utolsóként elkészülő munka befejezési időpontjának minimalizálása.

Példa

$F|perm|C_{max}$ (nincs előzés)

Munkák száma: $n = 5$

Munkák halmaza: $J = J_1, J_2, J_3, J_4, J_5$

Gépek száma: $m = 3$

Gépek halmaza: $M = M_1, M_2, M_3$

A lehetséges megoldások száma: $5! = 120$

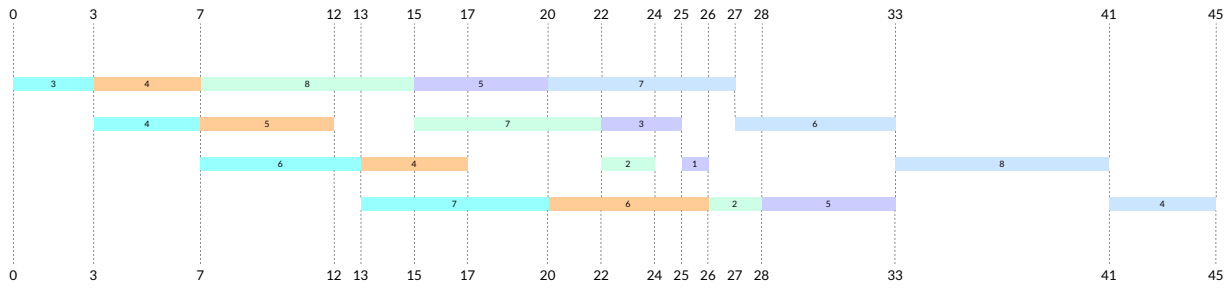
Egy lehetséges ütemterv: $[J_5, J_3, J_1, J_2, J_4]$, ekkor a végrehajtási sorrend minden gépen: $J_5 \rightarrow J_3 \rightarrow J_1 \rightarrow J_2 \rightarrow J_4$.

Az adott ütemterv szimulációja:
Adott egy feladat az alábbi időadatokkal:

$p_{i,j}$	M_1	M_2	M_3	M_4
J_1	3	4	6	7
J_2	4	5	4	6
J_3	8	7	2	2
J_4	5	3	1	5
J_5	7	6	8	4

Adott a következő ütemterv: $S = [J_5, J_3, J_1, J_2, J_4]$.

Induljunk az első gép első munkájától. Ábrázoljuk a munka végrehajtását időarányos szimbólummal (vonal vagy téglalap). Amikor az első munkát befejezte az első gép azonnal indul a második gép az első munka valamint az első gépen a második munka. Az első gép folyamatosan dolgozik, nem várakozik. Az első munka sem várakozik a gépek előtt. Egy közbenső gépen egy közbenső munka akkor indítható, ha a munka végrehajtása befejeződött az előző gépen és az aktuális gép befejezte az előző munkát. Ha a munka a korábbi gépen korábbban készült el mint ahogy az ektuális gép felszabadul, akkor a munka várakozik. Ha az aktuális gép szabadul fel korábbban, mint ahogy a munka megérkezik, akkor a gép várakozik.



1. ábra: A megoldás Gantt diagrammja

A befejezési időpontot az utolsó gép utolsó munkájának időpontja adja meg: $C_{max} = C_5 = 45$.

Ezeket a feladatokat inicializálja valamilyen véletlen szám generátorral! Legyen különböző méretűek:

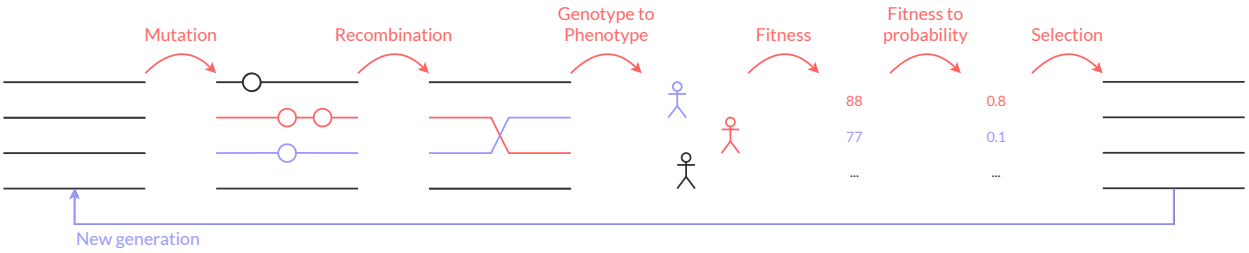
- a munkák száma legyen 10, 20, 50, 100, 200, 500,
- a gépek száma legyen 5, 10, 20.

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

Algoritmus: Genetikus Algoritmus

A genetikus algoritmusokat gyakran használjuk nevezetes problémák megoldására. Naív természeti megfigyeléseken alapszik. Egy

kromoszómákból álló populáción az alábbi műveleteket végezzük iteratívan:



1. Mutáció
2. Rekombináció
3. Genotípus →Fenotípus átalakítás
4. Fitness érték számítása
5. Túlélési valószínűség számítása
6. Túléléssel új generáció képzése

Az egyes kromoszómák a feladat lehetséges megoldásait tartalmazzák (értékek vektora). A mutáció valamilyen elemi (szomszédsági) művelet, melyel az egyes egyedeket (kromoszómákat) alakítjuk. A rekombináció során az egyedeket keresztezzük valamilyen módon, ezzel megtartva a jó tulajdonságokat. A genotípusból fenotípusba átalakítás során az egyes kromoszómáknak jelentést adunk (itt válik a feladat megoldásává). A fenotípus alapján kiszámoljuk a célfüggvényt, ez adja majd az egyed fitness értékét. A fitness értéket felhasználjuk arra, hogy egy valószínűséget számoljunk, ami azt adja meg, hogy az egyed milyen eséllyel éli meg a következő generációt. A kovetkező generáció a túlélő kromoszómákból áll (esetlegesen kívülről bekerülhetnek új, random elemek).

Az egyedek száma egy generációban lehet konstans és változó is. A mutáció és rakombináció során felül írhatjuk az egyes egyedeket, de hozzá is adhatjuk az eddigi elemekhez az újakat.

A mutációnak sokféle változata lehetséges. Ez lehet egyszerre többféle:

- n-opt
- Az egyes, véletlenszerűen választott szegmensek eltolása
- Az egyes, véletlenszerűen választott szegmensek megfordítása
- TSP és VRP esetén a mohó algoritmus alapján egy szegmens rendezése
- VRP esetén két útvonal közti csere
- Flow-shop esetén egy adott szegmens Palmer-index [1] alapján nem növekvő sorrend alapján való rendezése

[1]

$$I_j = -\sum_{j=1}^{m}\left(\frac{m-(2j-1)}{2}p_{i,j}\right)$$

- j - gép indexe
- m - gépek száma
- $p_{i,j}$ - i-edik munka munkaideje a j gépen

- Egy véletlenszerűen választott szegmens javítása egy lokális keresővel.

Ezeket egyszerre is használhatjuk, ahol valamilyen valószínűséggel választjuk ki az egyes operátorokat.

A keresztezés szintén lehet véletlenszerű és/vagy valamilyen heurisztikán alapuló. A keresztezendő egyedek kiválasztása is alapulhat heurisztikán és/vagy valamilyen véletlen szám generátoron.

A fitness érték számítása problémafüggő.

A túlélési valószínűség számítása lehet relatív:

$$P_i = \frac{f_i}{\sum_j f_j},\ f_i \geq 0$$

Lehet rendezés után valamilyen P_c, $0 < P_c < 1$ konstans megadásával. A különbség számítása lehet:

$$\begin{aligned} P_1 &= P_c \\ P_2 &= (1 - P_c) \cdot P_c \\ P_3 &= (1 - P_c)^2 \cdot P_c \\ P_{n-1} &= (1 - P_c)^{n-2} \cdot P_c \\ P_n &= (1 - P_c)^{n-1}, \end{aligned}$$

ahol a generáció egyedeinek száma n, a legjobb fitness értékű egyed az első, a legrosszabb az n-edik.

A túlélésbe bevehetjük az egyed egyedek „életkorát" is.

Az egyes paramétereket, mint a P_c, mutáció, rekombináció operátorainak valószínűségét nem feltétlen kell konstansként kezelni. Valamilyen stratégiával változhatnak. Ehhez érdemes megvizsgálni a Szimulált Hűtés hűtési stratégiáit.

Plusz feladat

Vizsgálja meg, hogy a populáció- és generációszám hogyan hat az eredményre és futási időre! Mikor érjük el azt a pontot, amikor már nem érünk el jobb eredményt új generációk bevezetésével.

Vizsgálja meg, hogy több mutációs és keresztezési operátor bevezetésével hogyan változik az eredmény!

Vizsgálja meg, hogy az egyes paraméterek időtől és/vagy fitnesstől függően, valamilyen stratégia alapján számítása hogyan hat az eredményre!

Az algoritmus végét egy ún. leállási feltétel vizsgálatával érjük el. Ez lehet egy adott generációszám elérése, egy előre meghatározott hőmérséklet elérése, valamennyi generáció 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.