# ÓRARENDGENERÁLÁS MEGOLDÁSI LEHETŐSÉGEI

**Molnárfi Brendon**

Programozó matematikus hallgató, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Tanszék

Cím: 3515 Miskolc, Miskolc-Egyetemváros, e-mail: aitnehez@uni-miskolc.hu

***Absztrakt***

*Ezen cikkben azokat az optimalizálási feladatokat fogom felvázolni, amelyeket megoldva középiskolai órarendet tudunk generálni. Használható eljárás a genetikus algoritmus, de előbb érdemes felbontani az eredeti négytényezős problémát egyszerűbb, kéttényezős és háromtényezős problémákra, melyek megoldhatóak hagyományos optimalizálási módszerekkel és együttesen alkotják a négytényezős optimalizálási feladat megoldását. Az ezek során kapott eredmények támpontokat adnak a genetikus algoritmus kapcsán és megállapíthatjuk, hogy a genetikus algoritmus a legjobb megoldás, amit használhatunk vagy sem.*

***Kulcsszavak:*** *órarend, optimalizálás, genetikus algoritmus*

***Abstract***

*In this article I will outline those optimalizational excersises, which need to solve to generate high school timetable. Genetic algorithm is an usable solution, but it's deserving to reduce the original 4-factored problem to simplier, 2-factored and 3-factored problems before that. Those are solvable with conventional optimalizational methods and make up together the solution of 4-factored problem. The results what we get so, give us strong points about genetic algorithm and we can determine, is genetic algorithm the best solution, we can apply or not.*

***Keywords:*** *timetable, optimalization, genetic algorithm*

# 1. Bevezetés

Az iskolák és egyetemek a különböző osztályok, tanárok és tantermek órarendjének elkészítéséhez ma már számítógépes segítséget, gyakran a mesterséges intelligencia egyik elterjedt eszközét, a genetikus algoritmust használják. Máskülönben nagyon időigényes, fáradtságos munka lenne összesen több mint 100 órarend ütközésmentes összehangolása, hiszen ugyanannak az osztálynak egy időben nem lehet több tanórája, egy tanár sem tud két helyen egyszerre jelen lenni és egy tanteremben sem lehet két tanóra egy időben. Nem is beszélve az olyan plusz feltételekről, amilyenek a termek kapacitása vagy az új osztályok képzésének szükségessége, a nyelvi vagy fakultációs órák miatt.

## 1.1 Kéttényezős feladat

Először egy kéttényezős problémát hozok létre azzal, hogy elhagyom a tanárokat és tantárgyakat, így csak az osztályokat kell hozzárendelni optimálisan a tantermekhez. Így kapunk egy úgynevezett halmazfelbontási feladatot, ami arról szól hogy az egyik halmaz (osztályok) és másik halmaz (tantermek) elemeihez 1:1 hozzárendelést kell elvégezni, vagyis minden osztályhoz pontosan 1 tanteremnek kell tartoznia és minden tanteremhez pontosan 1 osztálynak. Mindezt úgy hogy a legoptimálisabb megoldást kapjuk a kihasználatlanság minimalizálásának szempontjából. A kihasználatlanság az adott terem kapacitásának és az adott osztály létszámának a különbsége. Feltétel nyilván, hogy egy osztálynak nem lehet órája olyan teremben, amelynek a kapacitása kisebb az osztály létszámánál, a másik pedig, hogy egy évfolyam osztályai és az évfolyamon képzett nyelvi/fakultációs csoportok nem lehetnek benne egyazon megoldásban, vagyis egy adott időablakban. Mivel ebben az esetben könnyen előfordulhatna, hogy egyes diákoknak két helyen lenne jelenése egyidőben. Mint látható lesz, ezt úgy oldottam meg, hogy minden egyes nyelvi és fakultációs csoportot önálló osztályként kezeltem és a különböző évfolyamok nyelvi/fakultációs óráinak számításba vétele esetén különböző hozzárendeléseket végeztem. Így összesen 7 hozzárendelést kaptam, egyet arra az esetre, amikor kizárólag "alaposztályok" vannak, mivel mind a négy évfolyamon vannak nyelvi órák, így összesen négyet a nyelvi órákat is tartalmazó időablakokra és kettőt a fakultációs órákat is tartalmazó időablakokra, mert faktos órák csak a 11. és 12. évfolyamon vannak.

A feladat formalizálása:

- $n \in N$ : osztályok dbszáma
- $m \in N$ : tantermek dbszáma
- $l \in N^n$ : osztályok létszámai
- $k \in N^m$ : tantermek kapacitásai
- $h \in N^n$ : termek dbszámai, ahová az egyes osztályok befférnek
- $C \in N^{n \times m}$ : költségmátrix
- $A \in \{0;1\}^{n \times m}$ : párosításmátrix
- $X \in \{0;1\}^{n \times m}$ : hozzárendelés-mátrix

$$C_{ij} = \begin{cases} k_j - l_i, & ha\ X_{ij} = 1 \\ \infty, & egyébként. \end{cases}$$

$$A_{ij} = \begin{cases} 1, & ha\ l_i \leq k_j \\ 0, & egyébként. \end{cases}$$

$$X_{ij} = \begin{cases} 1, & ha\ az\ i.osztálynak\ a\ j.teremben\ lesz\ a\ tanóra \\ 0, & egyébként. \end{cases}$$

$$\sum_{j=1}^{m} C_j X_j \rightarrow min$$

$$\sum_{j=1}^{m} A_{ij} X_j = 1$$

$$n \leq m$$

$$i = 1, 2, \dots, n$$

$$j = 1, 2, \dots, m$$

Összes lehetséges esetek száma:

$$P = \prod_{i=1}^{n} h_i$$

Összes lehetséges megoldások száma:

$$P = \prod_{i=1}^{n} h_i - (i - 1)$$

A növekedési rend a következő: $\sim \Theta(2nm)$.

## 1.2 Háromtényezős feladat

A háromtényezős esetben osztályokat, tanárokat és tantárgyakat rendeltem egymáshoz. Ehhez a halmazlefedési feladatot alkalmaztam, ami abban különbözik a halmazfelbontásitól, hogy 1:N hozzárendelés van, ugyanis egy tanárhoz több osztályt és tantárgyat is rendelhetünk, sőt ugye kell is többet hozzárendelni. Mivel a hagyományos optimalizálási módszerek esetén két tényezővel tudunk dolgozni (ezért is lesz hasznos a mesterséges intelligencia nyújtotta optimalizálási módszer, a genetikus algoritmus használata), így a háromból két tényezőt összevontam. A tanárok és tantárgyak kettőse így 1 entitást alkot, ezáltal egy adott osztályt annyiszor hoztam létre, ahány tantárgy van az adott évfolyamon. A párosításmátrixban két feltételtől is függ, hogy 0 vagy 1 kerül a rublikába. Egyrészt, hogy az adott tanár tudja-e tanítani az adott tantárgyat, illetve hogy az adott osztálynak van-e ilyen tárgya (a 11. és 12. évfolyamon pl. nincs fizika, csak fakt van belőle). A minimalizálás pedig itt arra vonatkozik, hogy minél kisebb legyen a tanárok heti óraszámai közötti eltérés, ne fordulhasson elő, hogy mondjuk míg valaki 30 órát tart egy héten, addig más 5-öt. A futtatás után kapott eredményt látva megállapítható, hogy amennyire lehetett, sikerült kiküszöbölni a tanárok egyenlőtlen terhelését, megkaptuk a lehető legoptimálisabb osztály-tanár-tantárgy hozzárendelés-mátrixot, amely meghatározza, hogy egy adott osztálynak egy adott tantárgyat melyik tanár tartsa.

A feladat formalizálása:

- $n \in N$ : osztály-tantárgy kettősök száma
- $m \in N$ : tanárok száma
- $p \in N$ : tantárgyak száma
- $u \in N^p$ : az egyes tárgyakat hallgató osztályok dbszáma
- $v \in N^p$ : az egyes tárgyakat oktató tanárok dbszáma
- $t \in N^m$ : tanárok heti óraszámai
- $o \in N^n$ : osztály-tantárgy kettősökhöz tartozó heti óraszámok
- $A \in \{0;1\}^{n \times m}$ : párosításmátrix
- $X \in \{0;1\}^{n \times m}$ : hozzárendelés-mátrix

$$t_j = \begin{cases} t_j + o_i, & ha\ X_{ij} = 1 \\ t_j, & egyébként. \end{cases}$$

$$A_{ij} = \begin{cases} 1, & ha\ az\ i.osztály - tantárgy\ kettősben\ szereplő\ tárgyat\ tudja\ tanítani\ a\ j.tanár \\ 0, & egyébként. \end{cases}$$

$$X_{ij} = \begin{cases} 1, & ha\ az\ i.osztály - tantárgy\ kettősben\ szereplő\ osztálynak\ az\ ugyanitt\ szereplő \\ & tárgyat\ a\ j.tanár\ fogja\ tanítani \\ 0, & egyébként. \end{cases}$$

$$\sum_{j=1}^{m} |o_j X_j - avg(o)| \rightarrow min$$

$$\sum_{j=1}^{m} A_{ij} X_j = 1$$

$$i = 1, 2, \dots, n$$

$$j = 1, 2, \dots, m$$

$$k = 1, 2, \dots, p$$

Összes lehetséges esetek száma:

$$P = m^n$$

Összes lehetséges megoldások száma:

$$P = \prod_{k=1}^{p} v_k^{u_k}$$

Növekedési rend: $\sim \Theta(nm + nm^2)$.

## 1.2 Négytényezős feladat

A négytényezős feladat megoldása a két- és háromtényezős feladat megoldásainak egyesítéséből áll össze, vagypedig genetikus algoritmus megírása által. Akármelyiket is választjuk, az összes és lehetséges megoldások száma a háromtényezős feladatnál kapott szám lesz, mivel a kéttényezős feladat eset- és megoldásszámai nagyságrendileg elhanyagolhatóak ezekhez képest. A növekedési rend a két- és háromtényezős feladat megoldásainak egyesítése esetén a két növekedési rend összegeként áll elő.

A probléma leképezése a genetikus algoritmus összetevőire:

- **egyed:** időablak (pl. kedd 10-11 óra), melyben tanórák kerülnek megtartásra, különböző termekben, különböző tanárokkal, különböző osztályoknak. Az időintervallum nem kerül rögzítésre az egyes egyedeknél, mert ez esetben nem lehetne genetikus algoritmussal dolgozni egyrészt, másrészt szükség sincs rá, mert sorrendileg bármelyik időablak felcserélhető lesz bármelyik időablakkal, miután elértük a célt: az ütközésmentességet. Én Python nyelvet használok, ahol az implementáció szótár típus lesz, 3 adattaggal: kulcs szerepű azonosító (egész szám), az egyedhez aktuálisan tartozó gének, vagyis az időablakban levő tanórákra való referenciák (tömb), az optimumtól való eltérés, vagyis a büntetőfüggvények értékei (tömb).

- **populáció:** az egyedek összessége. Ez a heti összes időablakok száma lesz. Esetünkben hétfőtől szerdáig napi 8 időablakot állapítok meg, csütörtökre és péntekre napi 6-ot. Így lesz 36 fős populációnk, melyet listában tárolunk.
- **gén:** tanóra, ami egy osztály-tanár-tanterem-tantárgy négyes, plusz egy azonosító, melynek segítségével lehet hivatkozni a tanórára. Implementációja szótár. A gének összességét pedig listában tároljuk. Egy adott tantárgy egy időablakban többször is előfordulhat, viszont egy osztály, tanár, tanterem csak egyszer, ezért ezeknek a kapcsán büntetőfüggvényt tartunk számon. Ha többször is előfordul az adott időablakban mondjuk egy osztály, akkor az osztály-büntetőfüggvény értékét növeljük az előfordulások száma-1 értékkel, miután az összes osztályt végigvettük, kialakul a büntetőfüggvény végső értéke. Ugyanígy járunk el a tanárok és tantermek esetében is. Ezekhez az értékekhez a tanórák kulcs adattagján keresztül fér hozzá az időablak egyed.
- **célfüggvény:** a 3-féle büntetőfüggvény összesítésével kapott függvény, melynek optimális értéke 0. Csak ebben az esetben nincs ütközés. A büntetőfüggvények értékeit súlyozva vesszük számításba annak megfelelően, hogy az adathalmazban mik a darabszámok egymáshoz viszonyított arányai. Az én adathalmazomban 30 tanterem, 30 tanár és 60 osztály lesz, de utóbbira még külön ki kell térni. A 60 osztály között lesz 20 "alaposztály", 20 nyelvi és 20 fakultációs csoport. Ezáltal a tanterem- és tanárbüntetőfüggvény értéke 1,5-ös súllyal fog szorzódni az osztály-büntetőfüggvényhez képest. Miután összeadjuk őket, megkapjuk a célfüggvény értékét.
- **öröklődés:** egypontos keresztezéssel. A kromoszómák hossza, vagyis az egyedek génjeinek száma eltérő lehet, ezért maximálnunk kell, hogy az 1. és hanyadik gén között lehessen keresztezési pont, vagyis mi legyen a véletlenszám-generálás felső határa. Ezt a maximális értéket az adathalmaz méretének ismeretében határozzuk meg, és ez így egyben a keresztezés valószínűségi értékének meghatározására is alkalmas lehet.
- **mutáció:** ha egy új egyed valamely génjét mutálnunk kell, úgy oldjuk meg, hogy egy véletlenszerűen választott idegen egyed (vagyis nem szülő) sorrendileg utolsó génjét töröljük az idegen egyedből és ugyanakkor az egyedünk mutált génjévé tesszük.
- **szelekció:** rátermettség-arányos választással.

A feladat formalizálása:

- $P$ : populáció mérete
- $G$ : generációk száma
- $K$ : keresztezés valószínűsége
- $M$ : mutáció valószínűsége
- $n \in N$ : osztályok száma
- $m \in N$ : tanárok száma
- $p \in N$ : tantermek száma
- $o \in N^n$ : osztályok előfordulásainak száma
- $t \in N^m$ : tanárok előfordulásainak száma
- $h \in N^p$ : tantermek előfordulásainak száma
- $wt \in R^+$ : tanár-büntetőfüggvény súlya
- $wh \in R^+$ : tanterem-büntetőfüggvény súlya

- $b: N \rightarrow N$ : büntetőfüggvény

$$b(o) = \sum_{i=1}^{n} \max(o_i - 1, 0)$$

$$b(t) = \sum_{j=1}^{m} \max(t_j - 1, 0)$$

$$b(h) = \sum_{k=1}^{p} \max(h_k - 1, 0)$$

$$b(o) + wt * b(t) + wh * b(h) \rightarrow min$$

$$i = 1, 2, \dots, n$$

$$j = 1, 2, \dots, m$$

$$k = 1, 2, \dots, p$$

Növekedési rend: $\sim \Theta(G * P * \Theta(\text{célfüggvény}) * ((K * \Theta(\text{keresztezés})) + (M * \Theta(\text{mutáció}))))$

## 2. Összefoglalás

Utolsó lépésben még meghatározandó, hogy az egyes időablakokban mely osztályoknak milyen tárgy legyen tartva, annyit kell kiszűrni, hogy ugyanazon tanár ne fordulhasson elő többször egy időablakban, amit a foglalt tanárok tömbje/listája segítségével egyszerűen megtehetünk. A genetikus algoritmus leprogramozása és különböző méretű adathalmazokon való kipróbálása után minden kiderül a számunkra.

## 3. Köszönetnyilvánítás

A cikkben ismertetett kutatómunka az EFOP-3.6.1-16-2016-00011 jelű „Fiatalodó és Megújuló Egyetem – Innovatív Tudásváros – a Miskolci Egyetem intelligens szakosodást szolgáló intézményi fejlesztése” projekt részeként – a Széchenyi 2020 keretében – az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

### Irodalom

[1] Körei A.: https://www.uni-miskolc.hu/~matka/Dokumentumok/EP_feladatok.pdf. Miskolci Egyetem, Magyarország, 2020.

[2] D. Abramson, J. Abela: A parallel genetic algorithm for solving the school timetabling problem. Commonwealth Scientific and Industrial Scientific Organisation (CSIRO), Australia, 1991.

[3] F. G. Lobo, D. E. Goldberg, M. Pelikan: Time complexity of genetic algorithms on exponentially scaled problems. University of Illinois, USA, 2000.