Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 251-259 **https://doi.org/10.35925/j.multi.2020.2.30**

ÓRARENDGENERÁLÁS MEGOLDÁSI LEHETŐSÉGEI

Molnárfi Brendon
programtervező informatikus hallgató, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: molnarfi.brendon@student.uni-miskolc.hu

Nehéz Károly
egyetemi docens, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Tanszék
3515 Miskolc, Miskolc-Egyetemváros, Hungary, e-mail: aitnehez@uni-miskolc.hu

Absztrakt

Az órarendtervezés több évtized óta intenzíven kutatott tématerület, és mivel az NP-teljes feladatok témaköréhez tartozik, ezért a megoldásához heurisztikus eljárások szükségesek. Ebben a cikkben olyan optimalizálási megoldást fogunk bemutatni, amelyeket megoldva tantárgyi órarendet tudunk generálni. Használható eljárások a metaheurisztikus algoritmusok (pl. 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 metaheurisztikus algoritmusok kapcsán és megállapíthatjuk, hogy a metaheurisztikák a legjobb stratégiák vagy sem.

Kulcsszavak: *órarend, optimalizálás, metaheurisztikus algoritmusok*

Abstract

Timetable design has been an intensively researched topic for decades and since it belongs to the topic of NP-complete tasks, heuristic procedures are required to solve it. In this article I will outline those optimalizational excersises, which need to solve to generate high school timetable. Metaheuristic algorithms (for example genetic algorithm) are usable solutions, 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 metaheuristics and we can determine, is metaheuristic algorithms are the best ways, we can apply or not.

Keywords: *timetable, optimization, metaheuristic algorithms*

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 metaheurisztikákat (pl. genetikus algoritmust) használják. A heurisztikus algoritmusok nagyméretű problémákra is képesek jó eredményt adni rövid időn belül. A metaheurisztikus algoritmusok olyan heurisztikus algoritmusok, melyek nem csak egy adott típusú feladathoz használhatóak eredményesen. Az órarend generálási feladat során 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.

2. Metaheurisztikák alkalmazása az órarend generálási feladat megoldására

Ebben a fejezetben azt mutatjuk be, hogy az órarend generálási feladatra milyen metaheurisztikákat alkalmaztak a szakirodalomban. Az egyik legismertebb metaheurisztika a genetikus algoritmus, mely megoldások populációján hajt végre műveleteket (keresztezés, mutáció). Az [1] cikk szerzői egy olasz középiskola órarend generálási feladatot oldották meg genetikus algoritmussal. A szerzők egy sajátos mátrix reprezentációs módot alkalmaztak. Az órarend generálási feladatot egy 3D reprezentációs móddal is megoldhatjuk, erről a [2] cikk szerzői számolnak be. A reprezentációs mód az alábbi tényezőket tartalmazza: osztály, terem, nap, óra. A genetikus algoritmus során a géneket bitsorokkal és permutációkkal ábrázolták. Az órarendtervezést a [3] cikk szerzői szintén genetikus algoritmussal oldották meg. A cikkben az órarendtervezési feladat egy finomított változatát vették alapul. A szerzők kemény korlátokat (hard constraints) és könnyed korlátokat (soft constraints) definiáltak. A kemény korlátokhoz az alábbiakat sorolták: egy tanulóhoz (tankörhöz) egy időpontban csak egy kurzust rendelhetünk, a tanterem kapacitáskorlátját figyelembe kell venni az órák kiosztása során, a tantermek felszereltségét is figyelembe kell venni, és egyszerre csak egyetlen óra lehet egy teremben. A könnyed korlátnak az alábbiakat sorolták: a tanulóknak legalább egy kurzuson részt kell venni egy nap. A cikkben részletezésre kerülnek a genetikus algoritmus szelekciós és mutációs operátorai is. Egyéb könnyed korlátra is találunk példát a szakirodalomban, például a [4] cikk szerzői az oktató időbeosztásának preferenciáját egy ilyen korlátnak tekinti. Egy másik könnyed korlát az, hogy azonos tárgyból egy napon lehetőleg csak egy óra legyen.

A szimulált lehűtés [5-6], tabu keresés [7-8] szintén az órarend generálás során gyakran használt metaheurisztikák, mely egyetlen megoldáson hajtanak végre szomszédsági műveletet. Az órarendgenerálási feladatra a genetikus algoritmust választottam ki, így a továbbiakban a genetikus algoritmust mutatom be, majd a feladat kétényezős, háromtényezős és négytényezős modelljét vázolom.

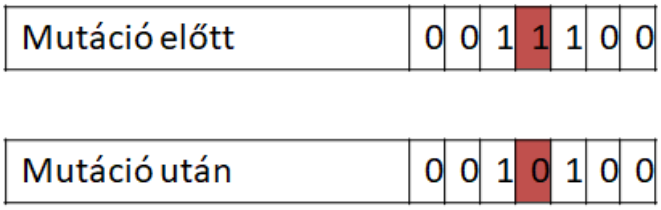
3. Genetikus algoritmus

A genetikus algoritmus egy metaheurisztikus algoritmus, mely megoldások populációját tartja fent. Az algoritmus (véletlenszerűen vagy konstrukciós heurisztikával generált) megoldások populációjából indul ki. A populáció a genetikus algoritmus során egyedekből áll, az egyed valójában egy-egy megoldást jelent. Maguk az egyedek pedig génekből állnak. A gének ábrázolására a leggyakoribb a bináris vagy permutációs ábrázolási mód. Minden optimalizálásnál fontos egy (vagy több) célfüggvényt is meghatározni. A célfüggvény maga az optimalizálás tárgya, amelynek a minimumát vagy maximumát keressük [9].

A genetikus algoritmus az új egyedeket mutációval és keresztezéssel állítja elő. A mutáció az egyed kismértékű változtatása. A keresztezés – mely az egyedek nagymértékű megváltozása - során pedig kettő (vagy több) szülő egyedből kettő (vagy több) gyerek egyed keletkezik. A leggyakoribb keresztezések bitsoros ábrázolásnál az egypontos és kétpontos keresztezések. Az egypontos keresztezés (1. ábra) lényege, hogy egyetlen keresztezési pontot határozunk meg. A keresztezési pont meghatározása leggyakrabban véletlenszerűen történik. A keresztezési pont két részre osztja az egyes szülők kromoszómáit. Az első gyerek az első szülő első kromoszóma darabját és a második szülő második kromoszóma darabját fogja megkapni, míg a második gyerek a második szülő első kromoszóma darabját, és az első

szülő második kromoszóma darabját kapja meg. A mutáció (2. ábra) az egyed olyan kismértékű változtatása, ahol legtöbbször csak egyetlen bit változik [10].

1. ábra. *Egypontos keresztezés.*



2. ábra. *Mutáció*

A genetikus algoritmus egy fontos lépése a szelekció. A szelekció határozza meg a szülők kiválasztását. Az alapelv az, hogy minél jobb szülőket próbáljon meg kiválasztani az algoritmus. A jó célfüggvény értékű egyedeket tehát nagyobb valószínűséggel választja ki a genetikus algoritmus. Ezt rátermettség arányos szelekciónak is nevezzük. A genetikus algoritmus pszeudo kódját a 3. ábra mutatja.

```
BEGIN
    1. Populáció inicializálása
    2. Populáció kiértékelése
    WHILE (kilépési feltétel nem teljesül) DO
        WHILE (a következő populáció elemei elő nem álltak) DO
            3. Kiválasztunk 2 egyedet az előző populációból (ezek lesznek a szülők)
            4. Két szülő egyed keresztezése (így létrejönnek a gyerek egyedek)
            5. A gyerek elemek mutációja
        END WHILE
    END WHILE
END
```

3. ábra. *Genetikus algoritmus pszeudo kódja*

4. 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 kellene lenniük egyszerre. 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 fakultációs órák csak a 11. és 12. évfolyamon vannak.

A feladat formalizálása:

- $n \in N$: osztályok darabszáma
- $m \in N$: tantermek darabszáma
- $l \in N^n$: osztályok létszámai
- $k \in N^m$: tantermek kapacitásai
- $h \in N^n$: termek darabszámai, 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

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

$$A_{ij} = \begin{cases} 1, & ha\ l_i \leq k_j \\ 0, & egyébként. \end{cases} \tag{2}$$

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

$$\sum_{j=1}^{m} C_j X_j \rightarrow min \tag{3}$$

$$\sum_{j=1}^{m} A_{ij} X_j = 1 \tag{4}$$

$$n \leq m$$

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

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

Összes lehetséges esetek száma:

$$P = \prod_{i=1}^{n} h_i \tag{5}$$

Összes lehetséges megoldások száma:

$$P = \prod_{i=1}^{n} h_i - (i - 1) \tag{6}$$

A szükséges számítási lépések növekedési rendje: $T(n, m) = \sim\Theta(2nm)$.

5. 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 fakultáció 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 száma
- $v \in N^p$: az egyes tárgyakat oktató tanárok száma
- $t \in N^m$: 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

$$t_j = \begin{cases} t_j + o_i, & ha\ X_{ij} = 1 \\ t_j, & egyébként. \end{cases} \tag{7}$$

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

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

$$\sum_{j=1}^{m} |o_j X_j - avg(o)| \rightarrow min \tag{8}$$

$$\sum_{j=1}^{m} A_{ij} X_j = 1 \tag{9}$$

$$i = 1, 2, \dots, n$$
$$j = 1, 2, \dots, m$$
$$k = 1, 2, \dots, p$$

Összes lehetséges esetek száma:

$$P = m^n. \tag{10}$$

Összes lehetséges megoldások száma:

$$P = \prod_{k=1}^{p} v_k^{u_k}. \tag{11}$$

A szükséges számítási lépések növekedési rendje: $T(n,m) = \Theta(nm + nm^2)$.

6. Négytényezős feladat

A négytényezős feladat megoldása a két- és háromtényezős feladat megoldásainak egyesítéséből áll össze, vagy pedig 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 formalizálása során már a genetikus algoritmusra támaszkodtam. A genetikus algoritmus az alábbi elemekkel rendelkezik:

- 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. A probléma megoldásához 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 tanár-büntetőfüggvény 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 genetikus algoritmus segítségével:

- 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

$$b(o) = \sum_{i=1}^{n} \max(o_i - 1, 0) \tag{12}$$

$$b(t) = \sum_{j=1}^{m} \max(t_j - 1, 0) \tag{13}$$

$$b(h) = \sum_{k=1}^{p} \max(h_k - 1, 0) \tag{14}$$

$$b(o) + wt \cdot b(t) + wh \cdot b(h) \rightarrow min \tag{15}$$

$$i = 1, 2, \dots, n$$
$$j = 1, 2, \dots, m$$
$$k = 1, 2, \dots, p$$

Számítási lépések növekedési rendje:

$$T(G, P, K, M) = \sim\Theta(G * P * \Theta(célfüggvény) * ((K * \Theta(keresztezés)) + (M * \Theta(mutáció))))$$

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.

7. Összefoglalás

A cikkben bemutatásra kerültek formálisan felírt órarendtervezési problémák és azok megoldásai. A két- és háromtényezős változat esetében a feladatok felírása alapján becsülni lehetett azon számítási lépéseknek a számát, amely során az összes lehetséges megoldás kiértékelésre kerül. A bonyolultság az osztályok és a tantermek számának függvényében volt megadható. A komplikáltabb, négytényezős változat esetében az optimalizálási probléma direkt megoldása már nem tűnt célszerűnek. Helyette metaheurisztikus algoritmusok (pl. genetikus algoritmus) alkalmazásával lehetett a problémát úgy reprezentálni, hogy az órarendek, mint egy populáció elemei jelenjenek meg. A jövőbeli kutatási irányom a

cikkben részletezett modell megvalósítása metaheurisztikákkal, az egyes algoritmusok hatékonyságának összevetése futási idő és fitnesz érték alapján.

Köszönetnyilvánítás

A cikkben ismertetett kutató munka 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] Colorni, A., Dorigo, M., Maniezzo, V.: *A genetic algorithm to solve the timetable problem.* Politecnico di Milano, Milan, Italy TR, 90-060. 1192. **https://doi.org/10.1007/978-3-642-77489-8_14**

[2] Sigl, B., Golub, M., Mornar, V.: *Solving timetable scheduling problem using genetic algorithms.* In Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003. (pp. 519-524). IEEE.

[3] Abdullah, S., Turabieh, H.: *Generating university course timetable using genetic algorithms and local search.* In 2008 Third International Conference on Convergence and Hybrid Information Technology (Vol. 1, pp. 254-260). IEEE. **https://doi.org/10.1109/ICCIT.2008.379**

[4] Jain, A., Jain, S., Chande, P. K.: *Formulation of genetic algorithm to generate good quality course timetable*. International Journal of Innovation, Management and Technology 2010, 1(3):248.

[5] Abramson, D.: *Constructing school timetables using simulated annealing: sequential and parallel algorithms.* Management Science 1991, 37(1):98-113. **https://doi.org/10.1287/mnsc.37.1.98**

[6] Aycan, E., Ayav, T.: *Solving the course scheduling problem using simulated annealing.* In 2009 IEEE International Advance Computing Conference pp. 462-466. IEEE. **https://doi.org/10.1109/IADCC.2009.4809055**

[7] Abdullah, S., Shaker, K., McCollum, B., McMullan, P.: *Construction of course timetables based on great deluge and tabu search*. In Metaheuristics Int. Conf., VIII Metaheuristic 2009, pp. 13-16.

[8] Islam, T., Shahriar, Z., Perves, M. A., Hasan, M.: *University timetable generator using tabu search.* Journal of Computer and Communications 2009, 4(16):28-37. **https://doi.org/10.4236/jcc.2016.416003**

[9] Genetikus algoritmus http://www.inf.u-szeged.hu/~jelasity/cikkek/mikonyv.pdf (Letöltés dátuma: 2020. 08. 27.)

[10] Aradi, P., Gräff, J., Lipovszki, G.: Számítógépes szimuláció https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0042_szamitogepes_szimulacio/ch05s02.html (Letöltés dátuma: 2020. 08. 27.)