FLOW-SHOP ÜTEMEZÉSI FELADATOKAT MEGOLDÓ GENETIKUS ALGORITMUS MUTÁCIÓ OPERÁTORAINAK ÉRZÉKENYSÉGVIZSGÁLATA

Oláh Béla
főiskolai tanársegéd
Szolnoki Főiskola, Műszaki és Agrárgazdálkodási Intézet, Műszaki és Gépészeti Tanszék, 5000 Szolnok, Tiszaligeti sétány, olahb@szolf.hu

Összefoglalás

Jelen tudományos munka célkitűzése egy általam már korábban elkészített és publikált permutáció flow-shop termelésütemezési feladatokat (FSSP) megoldó genetikus algoritmus (GA) mutáció operátorainak érzékenységvizsgálata. Dolgozatom az algoritmus által használt négy különböző mutáció operátor (fordított csere, 2 pontú inverzió, 1 és 2 pontú csere) összehasonlítására terjed ki a megoldások optimum-közeli hatékonyságának függvényében. Megvizsgálom, hogy az egyes mutációs eljárások adott arányánál hogyan változik a program teljesítménye, értékelem a kapott eredményeket és összefüggéseket keresek, melyek segítségével a genetikus algoritmus hatékonyabb alkalmazása lehetséges. Témaválasztásom gyakorlati jelentőségű eredménye annak kiderítése lesz, hogy milyen arányban érdemes az egyes mutáció operátorokat használni, a minél hamarabbi és minél inkább optimum-közeli megoldások szolgáltatása végett.

Kulcsszavak: *genetikus algoritmus, ütemezés, holtidő, mutáció, klónozás*

Abstract

The main goal of this scientific work is the sensitivity analysis of mutation operators of my own genetic algorithm (GA) for the permutation flow-shop scheduling problems (FSSP). This paper covers the comparison of the different mutation operators such as reciprocal exchange, simple inversion, swap and displacement used by the algorithm in function of the efficiency of the near optimal solutions. I analyze how the efficiency of the algorithm changes by some values of the mutation operators, I evaluate the obtained results and I search for relations that help to apply the GA more effectively and efficiently. The practical importance of my research results is to determine in what setting the mutation operators have to be used in order to supply near optimal solutions at the fastest possible time.

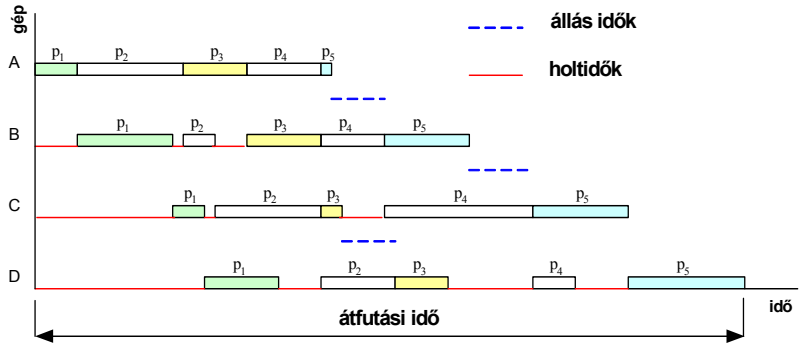
Keywords: *genetic algorithm, scheduling, idle time, mutation, cloning*

1. A Flow-shop ütemezési probléma megfogalmazása

Adott *n* számú termék, amelyeken *m* számú különböző munkafolyamatot kell elvégezni. A technológiai útvonal, ami az összes termékre nézve azonos, valamint a műveleti idők előre adottak. Meg kell határozni a termékeknek azt a sorrendjét a gépeken, amely bizonyos előre megadott szempontok szerint optimális (1. ábra).

A gyakorlatban is használt célfüggvények az alábbiak lehetnek:

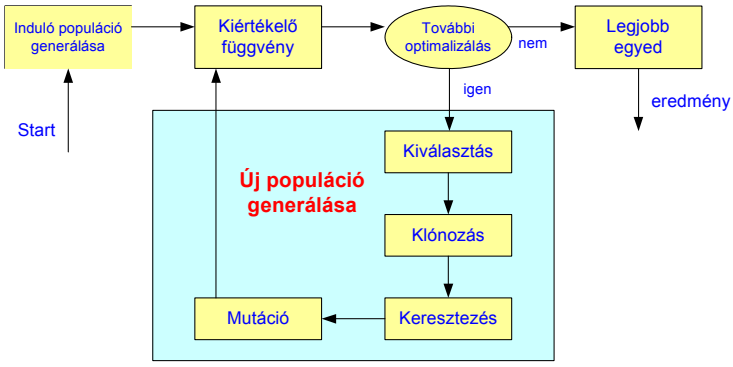
- minimális átfutási idő;
- technológiai berendezések maximális kihasználása (holtidők minimálása);
- minimális gyártásközi készletek (termékek állásidejének minimálása).



1. ábra. *Permutáció flow-shop ütemezés Gantt-diagramja*

2. A Genetikus algoritmus ismertetése

A genetikus algoritmusok fogalmát Holland [2] vezette be, tervezésük során az evolúciót tekinthetjük mintaképnek. Kezdetben nem optimálisan megírt, vagy paraméterezett programok a természetes kiválasztódás elve alapján fejlődnek, és közelítenek egy jó megoldáshoz. A genetikus algoritmus lényege, hogy rendelkezik a lehetséges megoldások egy populációjával, a populáción értelmezett a kiválasztási folyamat – amely az egyedek alkalmasságán alapul – és értelmezett néhány genetikus operátor (2. ábra).



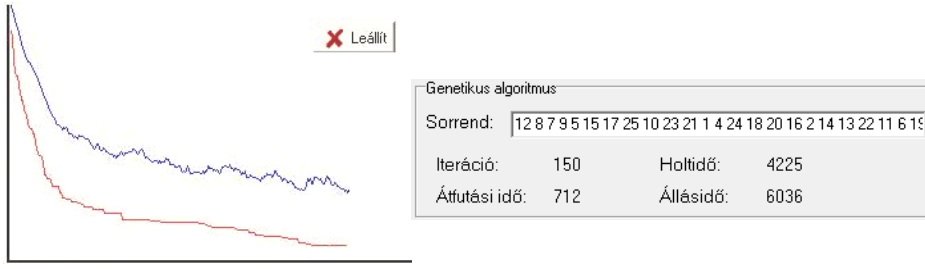
2. ábra. *A genetikus algoritmus folyamatábrája [10]*

A GA is – mint oly sok más a tudományban – a természettől kölcsönzött ötlet alapján működik [1]. Az életben évmilliók során kialakulnak azok az egyedek, amelyek legjobban alkalmazkodtak az élőhelyükhöz, amelyek fennmaradása biztosított. Ezek az egyedek genetikus állományukat – és ezzel jó tulajdonságaikat – továbbadják utódaiknak, biztosítva ezzel a populáció fennmaradását. Néha mutációk – véletlenszerű változások – adódnak a genetikus állományban. Az új egyedekben új tulajdonságok jelennek meg, amelyek vagy jobbak az eredetinél – és így az egyedek életben maradnak, tovább örökítve jó tulajdonságaikat –, vagy rosszabbak, s így elpusztulnak [3]. Ezt a folyamatot írtam át számítógépre a termelésütemezési problémák megoldására.

3. A számítógépes program bemutatása

A program felhasználói felületén a dialógusablakban a konstans jellegű paraméterek találhatók. Először beállítjuk a megmunkálni kívánt termékek, valamint a megmunkáló berendezések mennyiségét. Ezután a genetikus algoritmushoz szükséges alapadatokat állíthatjuk be tetszés szerint (populáció mérete, iterációk száma, a keresztezés, a mutáció és a klónozás aránya, kiválasztási stratégia). Az első fejezetben is említett optimalizáló célfüggvényt a megfelelő választógomb bekapcsolásával választhatjuk ki.

A genetikus algoritmus esetében egy kromoszóma a termékek tetszőleges sorrendjét jelenti, ez lesz az adatok helyes reprezentációja. A kiválasztódást alapbeállításban egyszerű fitnesz szerinti rendezéssel oldom meg, és a magasabb fitnesszel rendelkező egyedeket választom ki, de lehetőség van véletlenszerű, illetve a rulett-kerék mechanizmusnak megfelelő kiválasztásra. A GA megírása során két keresztezést *(CX* és *OX)* alkalmaztam. A program során a *reciprocal exchange (fordított csere)*, a *simple inversion (2 pontú inverzió)*, a *swap (1 pontú csere)* és a *displacement (2 pontú csere)* mutációkat használtam fel. Kiértékeléskor a maximális út megkeresésére alkalmas Bellmann-Pontrjagin-féle optimalizálási elvre épülő általam kidolgozott algoritmusba [5] történik a behelyettesítés.



3. ábra. *A genetikus algoritmus futási eredménye*

GA segítségével megoldva egy feladatot az iteráció előre haladtával a grafikonon nagyon szépen nyomon követhető a legjobb egyed célfüggvény szerinti

értéke valamint a populáció átlagértéke is (3/a. ábra). A legjobb egyed piros színnel (alsó görbe), míg az átlagérték kékkel (felső görbe) szerepel a grafikonon a jobb követhetőség érdekében. Mivel előfordulhat, hogy egy szülőt alacsonyabb fitnesz-értékű utód helyettesít, így a populáció átlagértéke emelkedhet is. Ezzel szemben a legjobb egyed fitnesz-értéke monoton csökkenő függvénnyel ábrázolható.

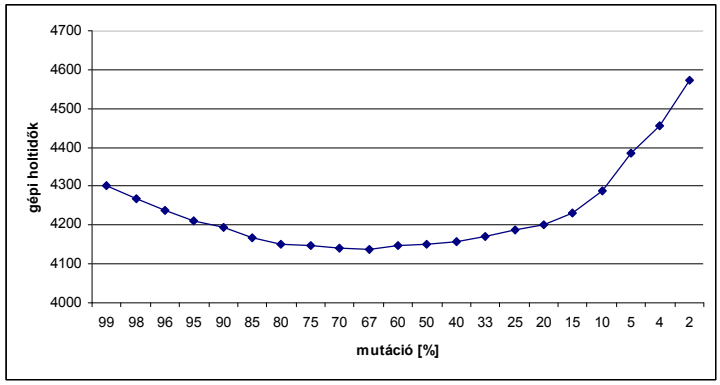
Az információs ablakban (3/b. ábra) az algoritmus futása közben folyamatosan kiírásra kerül az aktuális iterációk száma, valamint a legjobb egyedhez tartozó átfutási-, holt- és állásidők egyaránt. Az iterációk befejeztével megjelenik a legjobb egyedhez tartozó sorrend is. Az iteratív működés következtében több időre van szükségünk egy optimálishoz közeli megoldás eléréséhez, mint más heurisztikus esetben, viszont így lényegesen jobb eredmény érhető el. A feladat fitnesz-értékeit ábrázoló grafikonon megfigyelhető, hogy a GA végül olyan állapotba jutott, amelyben mind a legjobb individuum alkalmasság-értéke, mind pedig a populáció átlagos fitnesz-értéke a kezdeti rohamos javulás után beállt. A generációk számának növekedésével a legjobb egyed átlagos fitnesz-értéke egyre jobban megközelíti az optimális értéket, amely a vizsgált paramétertérben egy többé-kevésbé erős konvergenciát mutat. A programban lehetőség van a genetikus algoritmussal történő optimalizálás leállítására a leállít gomb megnyomásával. Ilyenkor a program megerősítést kér a felhasználótól az optimalizálás megszakítására. Az igen elfogadása után az információs ablakba kiíródnak az addig előállított legjobb egyed adatai. A nem gombon kattintva a közelítés tovább folytatódik a kívánt iterációig vagy egy újabb leállításig.

4. A mutáció operátorok összehasonlítása

Korábbi publikációimban [4, 6, 7, 8] már megvizsgáltam a termelésütemezési feladatokat megoldó különböző módszerek hatékonyságát, elvégeztem a genetikus algoritmus paraméter-érzékenységvizsgálatát, illetve összevetettem a szoftver által is kezelt keresztező operátorok eredményességét. Ezen dolgozatban a program által is használt négy mutáció operátor összehasonlítását tűztem ki célul.

Vizsgálataimat természetesen ugyanazon (esetemben 20 gépes, 25 termékes) permutáció flow-shop termelésütemezési feladatra végzem el, melyek alatt a keresztezés műveletet teljes egészében elhagyom, és csak a kiválasztott legjobb individuumok klónozásával valamint mutációjával állítom elő a soron következő populációt. Ezzel biztosítva, hogy az eredmények fejlődése csak a mutációnak legyen betudható, ami egyértelmű összehasonlítást tesz lehetővé.

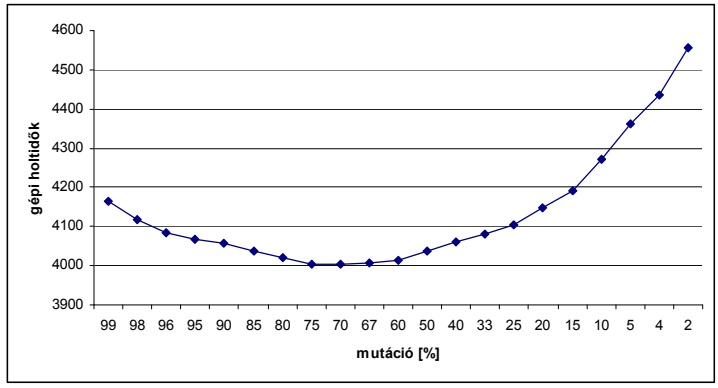
Első lépésben a program által használt négy mutáció operátor közül csak a simple inversion (2 pontú inverzió) mutációt alkalmazom. A 4. ábrán feltüntetett mutációs arányok (a populáció 99, 98, 96, …, 2%-a) mindegyikére a genetikus algoritmus által 30 futtatás után szolgáltatott célfüggvény (jelen esetben holtidő) értékek átlagai 100 egyedszámú populációt és a legjobb kiválasztási stratégiát alkalmazva 150 iteráció után a következőképpen alakultak:



4. ábra. *A 2 pontú inverzió holtidő-átlagai a mutáció arányának függvényében*

Az ábrából jól kivehető, hogy a mutáció arányának 80%-ig történő csökkenésével folyamatosan javulnak a holtidő-értékek, majd egy viszonylag konstans szakasz után (40-20% alatt) rohamos mértékű romlás következik be (a 2%-os mutáció már több mint 11%-kal eredményez rosszabb megoldásokat az optimumhoz képest). Megállapítható, hogy a genetikus algoritmus a 2 pontú inverzió mutációs operátor 80 és 40% közötti alkalmazása esetén eredményezte a legkisebb holtidő értékeket keresztezést nem használva. A minimumot a 67%-os mutáció (és ezáltal 33%-os klónozás) jelentette.

A következő vizsgálatban a program által kezelt négy mutáció operátor közül csak a fordított cserét (reciprocal exchange) használom. Az előzőhöz nagyon hasonló jellegű görbét kaptam. A vízszintes tengelyen az alkalmazott mutáció aránya, míg a függőlegesen továbbra is a holtidők szerepelnek.

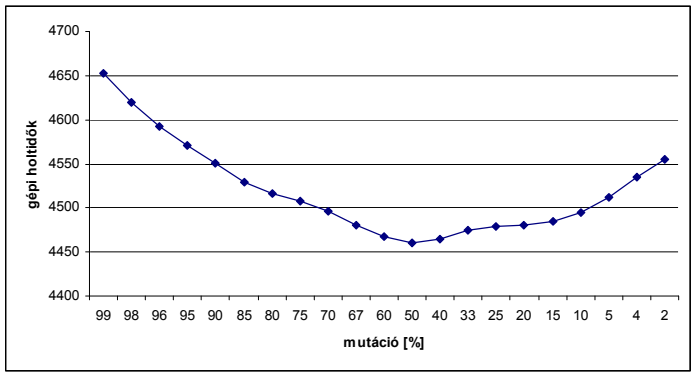


5. ábra. *A fordított csere holtidő-átlagai a mutáció arányának függvényében*

A diagramon jól látszik, hogy a mutáció arányának 75%-ig történő csökkenésével most is folyamatosan javulnak a holtidő-értékek, majd egy kezdeti lassú emelkedés után (25% alatt) rohamos mértékű hanyatlás következik be (a 2%-

os mutáció már közel 14%-kal eredményez rosszabb megoldásokat a minimumhoz képest). Kijelenthető, hogy a genetikus algoritmus a fordított csere mutációs operátor alkalmazása esetén a 75%-os aránynál produkálta az optimumot (4000).

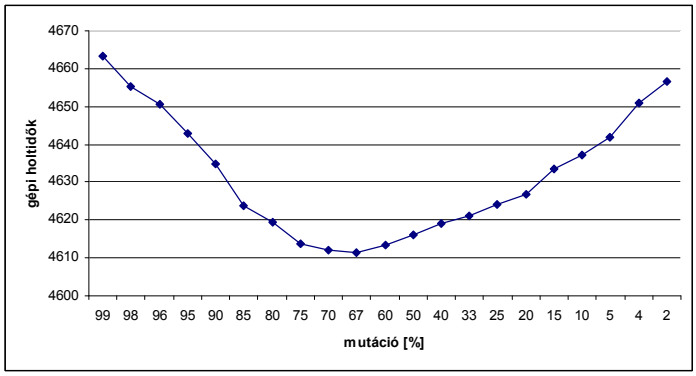
A következő esetben a mutáció operátorok közül csak az 1 pontú csere (swap) mutációt alkalmazom. Az eredményeket a 6. ábra szemlélteti:



6. ábra. *A 1 pontú csere által szolgáltatott holtidő-átlagok a mutáció függvényében*

Az eddigiektől kicsit eltérő jellegű görbén lassú javulás figyelhető meg a mutáció arányának egészen 50%-ot elérő csökkenéséig, majd ezután is csak szerény emelkedés tapasztalható. Azaz, a swap mutáció használata esetén épp a kisebb mutációs arányok jelentik a jobb megoldásokat. Bár nincs akkora eltérés az eredmények között, mint amekkorát a grafikon mutat, hisz a 99%-os mutáció is csak alig 4,5%-kal eredményez rosszabb megoldásokat a legkisebb értékűhöz (50% – 4461) képest.

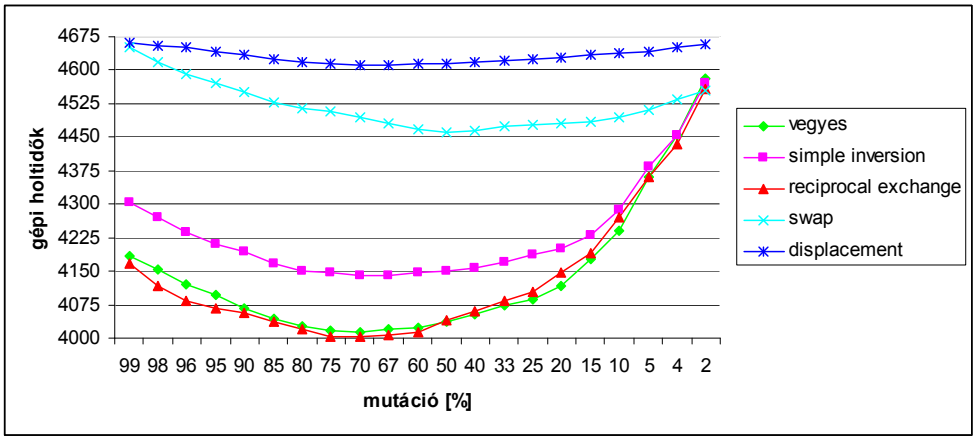
Az utolsó vizsgálatnál csak a 2 pontú csere (displacement) mutáció operátor segítségével állítja elő a következő populációt az algoritmus. A 7. ábrán feltüntetett mutációs arányok mindegyikére a GA által szolgáltatott holtidő értékek a következőképpen alakultak:



7. ábra. *A 2 pontú csere által szolgáltatott holtidő-átlagok a mutáció függvényében*

Most is szerény mértékű javulás látható a mutáció arányának egészen 67%-ot elérő csökkenéséig, majd ezután pedig még szerényebb emelkedés fedezhető fel a diagramon. A minimumot (4611) ugyan a 67%-os mutáció (és ezáltal 33%-os klónozás) jelentette, de a legrosszabb eredményhez (99%-os mutáció) képest is csak alig több mint 1%-kal ad jobb eredményt (az utóbbi két esetben csak a függőleges tengely korábbiaktól eltérő skálázása miatt tűntek meredekebbnek a görbék).

Végül megvizsgáltam, hogy a program által használt négy mutáció operátor egyenlő arányban történő alkalmazásakor hogyan alakul a grafikon. Kíváncsiságom oka az volt, hogy a keresztező operátorok összehasonlításakor [9] világossá vált, hogy a legjobb eredményt a GA az operátorok együttes alkalmazása esetén szolgáltatja, még az egyébként legjobb megoldásokat adó keresztezésnél is jobbat, azaz a gyengébbek nem hogy nem rontották a jobb operátorok teljesítményét, hanem még javították is azt. A jobb láthatóság és a már említett eltérő skálázásból adódó problémák kiküszöbölése végett egy diagramban ábrázoltam az eddigi és a vegyes alkalmazással kapott eredményeket.



8. ábra. *A különböző mutáció operátorok alkalmazásával kapott eredmények*

Az ábrán egyértelműen kitűnik, hogy a négy mutáció operátor közül kettő sokkal jobb eredményeket szolgáltat a másik kettőnél. A legjobb megoldásokat a fordított csere mutáció alkalmazása adja, de alig marad el tőle a 2 pontú inverzió operátor, ami csak átlag 2,5%-kal generál gyengébb célfüggvény-értékeket, sőt 10% alatti mutáció arányoknál a két görbe szinte fedi egymást. Ezzel szemben az 1 pontú csere közel 9,5%-kal rosszabb, mint a legkedvezőbb operátor, míg a 2 pontú csere már több mint 12%-kal, így kijelenthető, hogy ez utóbbi mutáció használata eredményezi a legnagyobb holtidőket.

Természetesen a jövőben érdemes elvégezni más – az operátorokat eltérő arányban használó – vegyes rendszerek esetén is ezt a vizsgálatot, hogy azok esetén hogyan alakulnának a diagramok.

5. Összefoglalás

A keresztező operátorok vizsgálatával ellentétben, most nem mondhatjuk el, hogy a mutációs eljárások együttes alkalmazásakor kapnánk a legjobb eredményeket, bár az operátorokat egyenlő arányban alkalmazó vegyes megoldás grafikonja szinte teljesen egyezik a legkisebb értékeket felvonultató fordított csere mutációéval. Az azonban már most is megállapítható, hogy inkább a fordított csere és a 2 pontú inverzió mutációkat érdemes preferálni az 1 és a 2 pontú cserével szemben a minél inkább optimum-közeli megoldások hatékony megtalálása végett.

6. Irodalomjegyzék

[1] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[2] Holland, J. H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology Control, and Artificial Intelligence*, The University of Michigan Press, 1975.

[3] Michalewich, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1996.

[4] Oláh, B.: *A flow shop ütemezési probléma optimalizálására szolgáló algoritmusok összehasonlítása*, OGÉT 2005 XIII. Nemzetközi Gépész Találkozó, Szatmárnémeti, 2005. pp. 268-272.

[5] Oláh, B.: *Genetic algorithm vs. Reinforcement learning*, Chapter 80 in DAAAM International Scientific Book 2009, Vol. 8, Published by DAAAM International, Vienna, Austria, 2009. pp. 831-844.

[6] Oláh, B.: *Genetikus algoritmus érzékenységvizsgálata*, Műszaki Tudomány az Észak-Alföldi Régióban 2010, Nyíregyháza, 2010. pp. 157-162.

[7] Oláh, B.: *Flow-shop termelésütemezési feladatokat megoldó genetikus algoritmus érzékenységvizsgálata*. Szolnoki Tudományos Közlemények XIV. Szolnok, 2010.

[8] Oláh, B.: *Sensitivity analysis of a genetic algorithm in function of the population size*. XXV. microCAD International Scientific Conference, University of Miskolc, Miskolc, Hungary, 2011. pp. 113-118.

[9] Oláh, B.: *Flow-shop ütemezést megoldó genetikus algoritmus keresztező operátorainak érzékenységvizsgálata*. XIX. Nemzetközi Gépész Találkozó – OGÉT 2011, Csíksomlyó, Románia, 2011. pp. 282-285.

[10] Pohlheim, H.: *Evolutionary Algorithms*, 2009., http://www.geatbx.com/docu/algindex.html