ALAP, HIBRID ÉS TÖBBSZINTŰ EVOLÚCIÓS ALGORITMUSOK

BASIC, HYBRID AND MULTILEVEL EVOLUTIONARY ALGORITHMS

Nagy Szilárd[*], *Dr. Jármai Károly*[**]

ABSTRACT

These methods are well used to solve nonlinear, multi dimensional engineering problems, where the usage of gradient based methods is difficult, or can't be used. In the last few years, research of these methods is got great emphasis. In this paper three basic algorithm - namely Random search, Firefly algorithm and Differential evolution -, hybrid- and multilevel methods of their combinations are introduced.

1. BEVEZETÉS

Az evolúciós algoritmusok természet inspirálta sztochasztikus, meta-heurisztikus kereső, optimáló eljárások. Jól használhatók nemlineáris, sok változós mérnöki feladatok megoldása során. Olyan esetekben is eredményre vezethetnek, ahol már a hagyományos gradiens alapú módszerek nehezen, vagy egyáltalán nem alkalmazhatóak, vagy nem adnak eredményt. A meta-heurisztikus algoritmusoknak két dologra van szükségük, a lehetséges megoldásokat kiértékelő függvényre, és az azokat generáló eljárásokra. E kettő megléte esetén már képes lehetséges optimumot találni. Természetükből adódóan ritkán vagy egyáltalán nem mondható meg, hogy a kapott megoldás lokális, vagy globális optimum e? Biztosan csak az állítható, hogy a kapott eredmény jobb, mint a kiindulási állapot.

Amit az ember gondolkodással próbál megoldani, arra a természet sok esetben talál a változatosság, és a szelekció eszközeit felhasználva hatékonyabb megoldásokat. A külső hatásokra adott véletlenszerű válasz (mutáció), eredményezi az egyedek sokszínűségét. A túlélésért folytatott harcban, az élelem után való folyamatos kutatásban stb. az életképesebb egyedek vagy kiválogatódnak, tulajdonságaik tovább örökítésének lehetőségét magukban hordozva. A biológiai szaporodást, mutációt és szelekciót alkalmazzák a különböző Genetikus algoritmusok (GA) [1], vagy az E.Coli baktériumok szaporodása (BFOA) [2]. Élelem keresési stratégián alapul az egyik legrégebben publikált algoritmus a Hangya kolónia optimalizáció (ACO) [3], vagy a méh algoritmus (ABC) [4,5,6]. A felsoroltakon kívül még rengeteg, az előzőekhez hasonló algoritmus létezik és van használatban. A teljesség igénye nélkül néhány ezek közül: Részecske csoport módszer (PSO) [7], Denevér algoritmus (BATA) [8], Kakukk keresés (CS) [9], Kulturális algoritmus (CA) [10].

Az evolúciós algoritmusok a keresési teret részhalmazokra, populációkra bontják. A populációk egyedei egy-egy állapotot, lehetséges megoldást reprezentálnak. Az egyedek tovább bonthatók tulajdonságokra, melyek a tervezési változóknak feleltethetők meg. Az egyedek mutáció, keresztezés és szelekció ismételt alkalmazásával fejlődnek generációról - generációra. Matematikailag egy-egy vektorral fejezhető ez ki

$$\tilde{x}_i^{(G)} = \left(x_{i,1}^{(G)}, x_{i,2}^{(G)}, \dots, x_{i,n}^{(G)}, \dots, x_{i,D}^{(G)}\right) \quad \forall i \in [1; NP] \tag{1}$$

ahol G az adott generáció, D a célparaméterek száma és NP a populáció mérete. Az egyedekről általánosságban elmondható, hogy

$$\forall x_{i,n}^{(G)} \in \mathbb{R} \cap \left[x_{lb,n}; x_{ub,n}\right] \tag{2}$$

ahol $x_{lb,n}$ és $x_{ub,n}$ az $n-dik$ egyed (tervezési paraméter, függvény változó, stb.) alsó és felső határa.

* *PhD hallgató, Miskolci Egyetem; szerszám és készülék tervező mérnök, Aventics Hungary Kft.*
** *egyetemi tanár, Miskolci Egyetem*

2. VÉLETLEN KERESÉS

A Véletlen keresés (RS) az egyik legegyszerűbb meta-heurisztikus optimáló algoritmus. A szó szoros értelmében nem tekinthető evolúciós módszernek, mert semmilyen biológiai analógiát nem tartalmaz. Teljesen véletlenszerű, hogy generációnként talál-e jobb megoldást. E tulajdonsága miatt optimálásra önmagában nem is szokták használni, hanem csak kezdeti populáció generálásához.

Az RS a keresési térben véletlenszerűen választ egy új pozíciót. Ha az új pozícióhoz tartozó fitnesz érték jobb az előzőnél, akkor megtartja, egyébként nem frissíti az aktuális egyedet.

$$x_{i,j}^{(R)} = \left(x_{ub,j} - x_{lb,j}\right) rand(0,1) + x_{lb,j} \tag{3}$$

$$x_{i,j}^{(G+1)} = \begin{cases} x_{i,j}^{(R)} & ha\ f\left(x_i^{(G+1)}\right) < f\left(x_i^{(G)}\right) \\ x_{i,j}^{(G)} & egyébként \end{cases} \tag{4}$$

3. DIFFERENCIÁLIS EVOLÚCIÓ (DE)

A Differenciális evolúció egy a klasszikus eljárások közül. Először 1990-es években Price és Storn mutatta be [11]. Az alap ötlet, hogy a populáció generációnkénti fejlődéséhez vektor különbségeket használ. Ez az eddig használt genetikus algoritmusokhoz képest nagy számítási teljesítmény javulást eredményezett, mert a populáció egyedei valódi alakjukban szerepelnek és nincs szükség komplikált kódolásra.

A DE önhivatkozó reprodukciós sémája eltér más evolúciós algoritmusokéitól. Az első generációt leszámítva, az adott populáció, illetve annak egyedei véletlenszerűen kombinálásra kerülnek, hogy kialakítsák a következő populációt az alábbi lépesek szerint:

3.1. Mutáció

Teljesség igénye nélkül néhány mutációs stratégia:

$$\bar{v}_i^{(G)} = \bar{x}_{r1}^{(G)} + F\left(\bar{x}_{r2}^{(G)} + \bar{x}_{r3}^{(G)}\right) \tag{5}$$

$$\bar{v}_i^{(G)} = \bar{x}_{best}^{(G)} + F\left(\bar{x}_{r1}^{(G)} + \bar{x}_{r2}^{(G)}\right) \tag{6}$$

$$\bar{v}_i^{(G)} = \bar{x}_i^{(G)} + F\left(\bar{x}_{best}^{(G)} + \bar{x}_i^{(G)}\right) + F\left(\bar{x}_{r1}^{(G)} + \bar{x}_{r1}^{(G)}\right) \tag{7}$$

$$\bar{v}_i^{(G)} = \bar{x}_{best}^{(G)} + F\left(\bar{x}_{r1}^{(G)} + \bar{x}_{r2}^{(G)}\right) + F\left(\bar{x}_{r3}^{(G)} + \bar{x}_{r4}^{(G)}\right) \tag{8}$$

$$\bar{v}_i^{(G)} = \bar{x}_{r1}^{(G)} + F\left(\bar{x}_{r2}^{(G)} - \bar{x}_{r3}^{(G)}\right) + F\left(\bar{x}_{r4}^{(G)} - \bar{x}_{r5}^{(G)}\right) \tag{9}$$

ahol $r_1, r_2, r_3, r_4, r_5 \in [1; NP]$ egymástól független véletlen egész szám, $F > 0$ valós konstans (lépték tényező) és x_{best} az eddig talált globális minimum helye. A leggyakrabban használt stratégiák 12 az eredeti (5) és (7)

3.2. Keresztezés

$$u_{i,j}^{(G)} = \begin{cases} v_{i,j}^{(G)} & ha\ rand(0,1) < CR\ vagy j = j_r \\ x_{i,j}^{(G)} & egyébként \end{cases} \tag{10}$$

ahol $CR \in [0,1]$ keresztezési arány, j_r véletlen egész szám. A CR értékét a gyakorlatban gyakran 0,9-re választják 12.

3.3. Szelekció

$$\bar{x}_i^{(G+1)} = \begin{cases} \bar{u}_i^G & ha\ f\left(\bar{u}_i^{(G)}\right) < f\left(\bar{x}_i^{(G)}\right) \\ \bar{x}_i^G & egyébként \end{cases} \tag{11}$$

Megvizsgálja, hogy az új egyed fitnesz értéke jobb-e az aktuálisnál. Ha igen, megtartja, egyébként elveti. Ez a változás biztosítja, hogy a módszer folyamatosan tartson egy jobb megoldás felé.

Az eredeti DE algoritmusban az F és CR paraméterek hasonlóan az NP-hez a keresési folyamat során nem változnak. Azóta több kutatás is javasolta ezek futásidejű adaptív változtatását. F-re és CR-re ad módszert a [12], a populáció méretére pedig [13,14].

A differenciális evolúció pszeudokódját az 1. algoritmus szemlélteti. A sorok között könnyen felismerhető a korábban említett három lépés. Mutáció 4. sor, keresztezés 5-12. sorok és végül a szelekció 13-17. sor. A kezdeti populáció inicializálás történhet bármilyen eloszlású véletlen szám alapján, de ajánlott az egyenletes eloszlású.

1. P populáció inicializálása
2. **while** *kilépési feltétel nem igaz* **do**
   for $i = 1;\ i \leq NP;\ i = i + 1$ **do**
   $\bar{v}_i^G$ mutációs vektor előállítása
   $j_{rand} \in [1; D]$ véletlen egész szám

for $j = 1;\ j \le D;\ j = j + 1$ **do**
if $rand(0,1) < CR$ **then**
$u_{i,j}^{G} = v_{i,j}^{G}$
else
$u_{i,j}^{G} = x_{i,j}^{G}$
end
end
if $f\left(\bar{u}_i^{(G)}\right) < f\left(\bar{x}_i^{(G)}\right)$ **then**
$\bar{x}_i^{(G+1)} = \bar{u}_i^{(G)}$
else
$\bar{x}_i^{(G+1)} = \bar{x}_i^{(G)}$
end
end
3. end

1. algoritmus DE pszeudokódja

4. SZENTJÁNOSBOGÁR ALGORITMUS (FA)

A Szentjánosbogár algoritmust először [15] javasolta 2009-ben. Napjainkban az egyik legígéretesebb rajintelligencián alapuló optimalizáló eljárás [16], a szentjánosbogarak idealizált szociális viselkedést imitálja.

1. Minden szentjánosbogár uniszex, így egy bogár vonzza az összes többit függetlenül a nemétől.
2. Vonzás arányos a fényerősséggel és függ a közöttük lévő távolságtól. Két világító szentjánosbogár közül a kevésbé fényes elmozdul a fényesebb felé.
3. Fényességet a célfüggvény fitnesz értéke befolyásolja.
4. Ha nem talál saját magánál fényesebb bogarat akkor véletlen mozgásba kezd.

A módszer nagyon hasonlít a Részecske csoport (PSO) algoritmushoz, minél jobb megoldást talál az egyed, annál erősebb fényt bocsájt ki, ami oda vonzza a populáció többi egyedét az adott területre.

A legegyszerűbb esetben egy szentjánosbogár I fényessége x pontban választható $I \propto f(x)$ alakban. A β vonzás azonban relatív, minden egyed más és másképpen érzékeli. Így hát meghatározásánál figyelembe kell venni az i és j bogár közötti r_{ij} távolságon túl, a közeg által elnyelt fény mennyiségét is.

$$\beta(r) = \beta_0 e^{\gamma r^m} \tag{12}$$

ahol β_0 a vonzerő $r = 0$ távolságnál, γ pedig konstans. A költségesen számolható exponenciális függvény helyett használható a sokszor gyorsabban kiszámítható (13) alak is.

$$\beta(r) = \frac{1}{1 + r^2} \tag{13}$$

Távolság a i és j egyed között Euklideszi távolságként definiálható

$$r_{ij} = \left\|\bar{x}_i^{(G)} - \bar{x}_j^{(G)}\right\| = \sqrt{\sum_{k=1}^{D}\left(x_{i,k}^{(G)} - x_{j,k}^{(G)}\right)^2} \tag{14}$$

ahol $x_{i,k}^{(G)}$ a $G.$ generáció i szentjánosbogár térbeli koordinátájának k komponense.

Az i egyed új pozíciója, mozgása egy fényesebb j szentjánosbogár felé

$$\bar{x}_i^{(G+1)} = \bar{x}_i^{(G)} + \beta\left(r_{ij}\right)\left(\bar{x}_j^{(G)} - \bar{x}_i^{(G)}\right)r + \alpha(rand(0,1) - 0{,}5) \tag{15}$$

ahol az egyenlet második fele egy α konstans paraméterrel arányos véletlen mozgás. Az előzőeket foglalja össze 2. algoritmus.

1. P populáció inicializálása
2. α, β_0, γ inicializálása
3. I_i fényerősség kiszámítása $\bar{x}_i^{(G)}$ pontban
4. **while** *kilépési feltétel nem igaz* **do**
for $i = 1;\ i \le NP;\ i = i + 1$ **do**
for $j = 1;\ j \le NP;\ j = j + 1$ **do**
if $I_j < I_i$ **then**
kiszámítása (14) alapján
szentjánosbogár mozgatása
I_i frissítése
end
end
if *nem történt mozgás* **then**
véletlen mozgás
end
end
5. end

2. algoritmus FA pszeudokódja

1. táblázat Szabványos teszt függvények

Név	Formula	Keresési tér	Globális minimum
Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)+\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right)+20+e$	$x_i \in [-32.768,32.768]\ i=1,2,\ldots,D$	$f(x^*)=0,\ x^*=(0,0,\ldots,0)$
Griewank	$\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$x_i \in [-600,600]\ i=1,2,\ldots,D$	$f(x^*)=0,\ x^*=(0,0,\ldots,0)$
Levy	$\sin^2(\pi\omega_1)+\sum_{i=1}^{D-1}(\omega_i-1)^2(1+10\sin^2(\pi\omega_i+1))+(\omega_D-1)^2(1+\sin^2(2\pi\omega_D))$ where $\omega_i=1+\frac{x_i-1}{4}\ i=1,2,\ldots,D$	$x_i \in [-10,10]\ i=1,2,\ldots,D$	$f(x^*)=0,\ x^*=(1,1,\ldots,1)$
Rastrigin	$10D+\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)\right)$	$x_i \in [-5.12,5.12]\ i=1,2,\ldots,D$	$f(x^*)=0,\ x^*=(0,0,\ldots,0)$
Rosenbrock	$\sum_{i=1}^{d-1}\left(100(x_{i+1}-x_i^2)\right)^2+(x_i-1)^2$	$x_i \in [-5,10]\ i=1,2,\ldots,d$	$f(x^*)=0,\ x^*=(1,1,\ldots,1)$
Sphere	$\sum_{i=1}^{d}x_i^2$	$x_i \in [-5.12,5.12]\ i=1,2,\ldots,d$	$f(x^*)=0,\ x^*=(0,0,\ldots,0)$
Zakharov	$\sum_{i=1}^{d}x_i^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^4$	$x_i \in [-5,10]\ i=1,2,\ldots,d$	$f(x^*)=0,\ x^*=(0,0,\ldots,0)$

5. TÖBB SZINTŰ ÉS HIBRID ALGORITMUSOK

A bemutatott alap eljárások önmagukban is hatékonyak, de egymással kombinálva ez a hatékonyság növelhető. Hibrid algoritmus képezhető, ha több eljárás ugyanazon populáció egy-egy részén párhuzamosan, vagy kvázi párhuzamosan dolgozik. Ezzel kiegészítve egymás hatékonyságát, növelve a konvergencia sebességét.

A FA és DE algoritmusnak is meg van a saját előnye, mindkettő széles körben használható optimálási problémák megoldására. A [17] javaslatot tesz egy hibrid algoritmusra, melyet hFADE-nek hív. A javasolt módszer egyesíti az FA és DE előnyeit. A FA algoritmus egy pontba vonzási mechanizmusával, valamint a DE keverési és szétszórási képességével, megnöveli a konvergencia sebességét és ugyanakkor az egyedek sokféleségét.

Az algoritmusok sok összetevője közül az intenzifikálás és a diverzifikálás (más néven a feltárás és a kiaknázás) a két legfontosabb jellemzője. A keresési terület globális szinten történő feltárásához szükséges egy jól működő diverzifikációs, feltárási stratégia. Az intenzifikációs, kiaknázási stratégia pedig a lokális keresésben segíti az egyedeket. Az algoritmusok pontossága, és sebessége növelhető e két képesség folyamatos egyensúlyban tartásával. A fenti mechanizmus eléréséhez a két eljárás kombinációját a 3. algoritmus foglalja össze.

A hibrid technológián kívül másik elgondolás lehet a kombinációra a többszintű algoritmusok alkalmazása. A [18] végzet kutatásokat ilyen téren (mRSFA). Az alap elgondolás, hogy az iterációs lépések első részében durva, és gyors keresést alkalmaznak a teljes keresési téren egy gyors algoritmussal. A választott gyors algoritmus az RS, mivel könnyen implementálható és a függvény értékek kiszámításán túl nem igényel semmilyen más kiegészítő számítást.

Az optimálás második felében egy lassú, de jól konvergáló és sok esetben bizonyított eljárás kapja meg a populációt (FA). Ez már az előzőnél jóval lassabb módszer, mivel egy iterációs lépésen belül, komoly kiegészítő számításokat is igényel, és a függvény értéket legrosszabb esetben NP^2-szer számolja ki. A mRSFA lépéseit a 4. algoritmus szemlélteti.

1. P populáció inicializálása
2. P véletlenszerű kettéosztása P_{DE}ésP_{FA}
3. **while** *kilépési feltétel nem igaz* **do**
   FA algoritmus végrehajtása P_{FA}-n
   DE algoritmus végrehajtása P_{DE}-n
   globális minimum frissítése
   P_{FA}, P_{DE} újra osztása véletlenszerűen
4. **end**

3. algoritmus hFADE pszeudokódja

1. P populáció inicializálása
2. **while** 1. *kilépési feltétel nem igaz* **do**
   for$i = 1;\ i \leq NP;\ i = i + 1$**do**
   új egyed generálása (3) alapján
   globális minimum frissítése
   end
3. **end**
4. **while** 2. kilépési feltétel nem igaz **do**
   új egyedek generálása FA szerint
   globális minimum frissítése
5. **end**

4. algoritmus mRSFA pszeudokódja

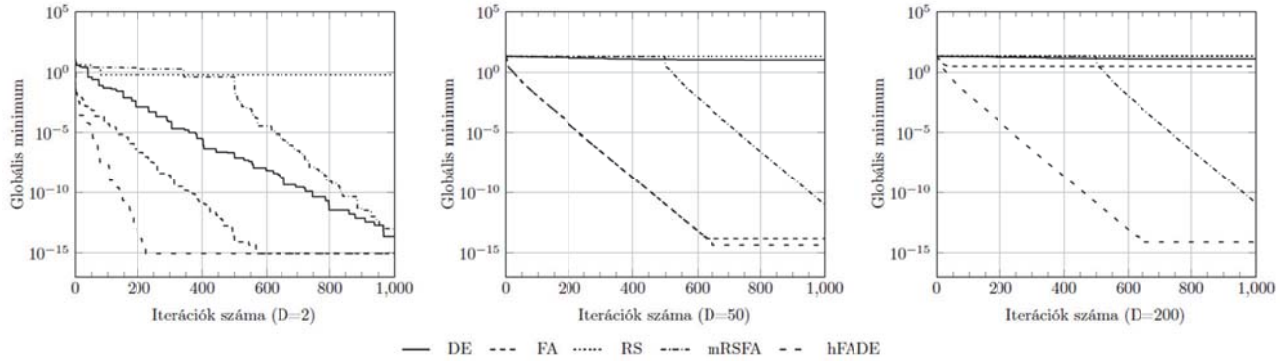
A [18] eredményei alapján a konvergencia sebességben és megbízhatóságban nincs nagy különbség az alap Szentjánosbogár algoritmushoz képeset. A futási idő, célfüggvény kiszámítási igénye viszont nagyban lecsökken az alap FA-hoz képest. A 18 végzett teszt alapján akár 50% - 90%-al is gyorsabb lehet kétváltozós függvények esetében. Természetesen ez nagyban függ a vezérlő paramétertől és magától a problémától is.

6. TESZT FÜGGVÉNYEK ÉS SZIMULÁCIÓ

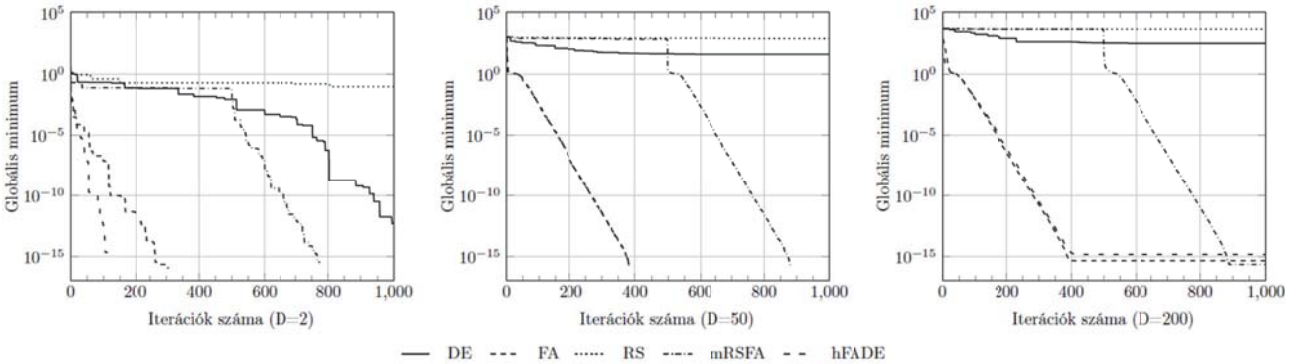
A teszt függvények hasznosak az új és már meglévő evolúciós módszerek jellemzőinek - konvergencia sebessége, precizitás, hatékonyság stb. – kiértékelésére, összehasonlítására. A cikkben bemutatott algoritmusok hét szabványos függvény használatával kerülnek összehasonlításra. Ezeket a függvényeket az 1. táblázat foglalja össze. A függvények között vannak könnyen optimálható monoton és nehezebb, az optimum környezetében ellaposodó és sok lokális minimumot tartalmazó függvények. Utóbbiak globális minimumának a meghatározása általában nehezebb, mert ha a keresés során lokális minimum közelébe kerül az eljárás, onnan nehezen, vagy egyáltalán nem tud kijönni és jobb megoldást keresni.

A szimulációk azonos körülmények között kerültek elvégzésre, ami az iterációk számában és populáció méretében tükröződik. Az iterációk száma 1000, populáció mérete pedig függetlenül az optimálandó paraméterek számától mindig $NP = 100$ egyed. A szakirodalmakban megtalálható legtöbb összehasonlítás általában kevés paraméterrel történik. Ettől a szokástól eltérően itt három különböző méretű paraméterszám kerül vizsgálatra. Az általánossá vált $D = 2$-n kívül a $D = 50$ és már nagynak számító $D = 200$.

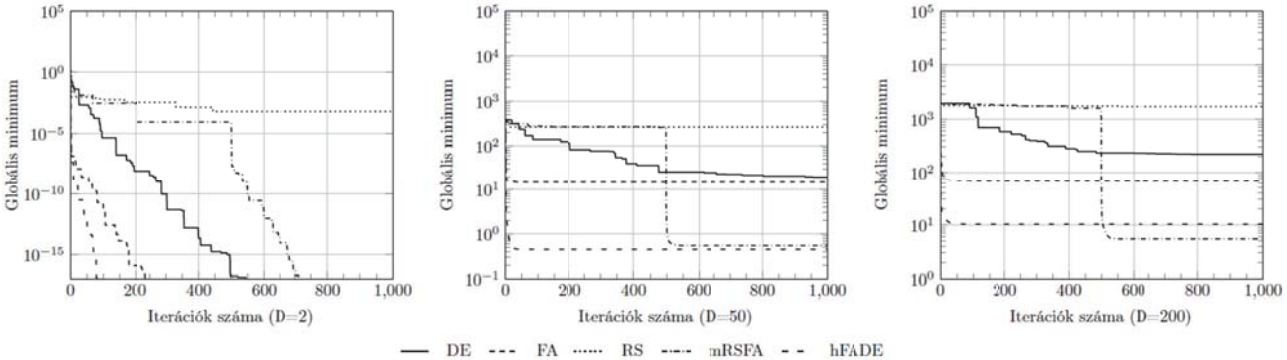
Az algoritmusok konstansai pedig megegyeznek a már korábban említett szakirodalmak ajánlásaival. A Differenciál evolúció esetén $F = 0{,}5$ és $C_r = 0{,}9$. A Szentjánosbogár esetén pedig $\beta_0 = 2$, $\gamma = \frac{1}{S^2}$ ahol S a változók átlagos tartománya, és $\alpha = 0{,}2 * 0{,}95^G$. A hibrid és több szintű algoritmusok konstansai megegyeznek az alap algoritmusoknál használtakkal. A hFADE esetén nem található szakirodalmi ajánlás, hogy milyen arányban érdemes szétosztani a populációt a két eljárás között. Itt 50% - 50%-ban lettek az egyedek szétosztva. A mRSFA esetén is az arany középúttal készült a szimuláció. Az iterációs lépések 50 %-ban dolgozik az RS és 50 %-ban az FA.



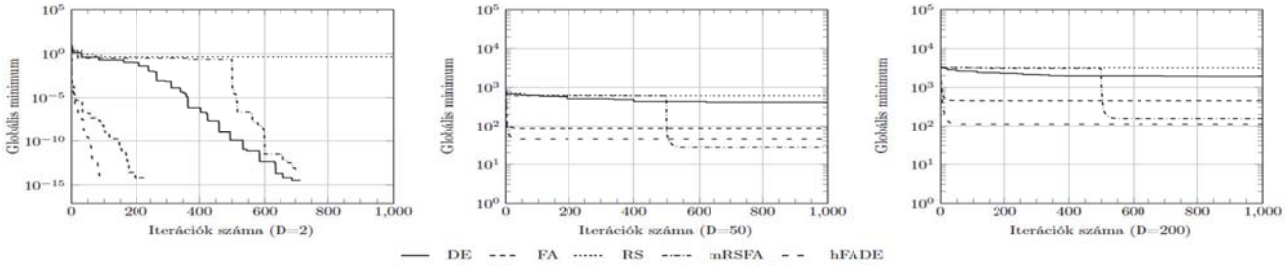
1. ábra Konvergencia Ackley függvény esetén



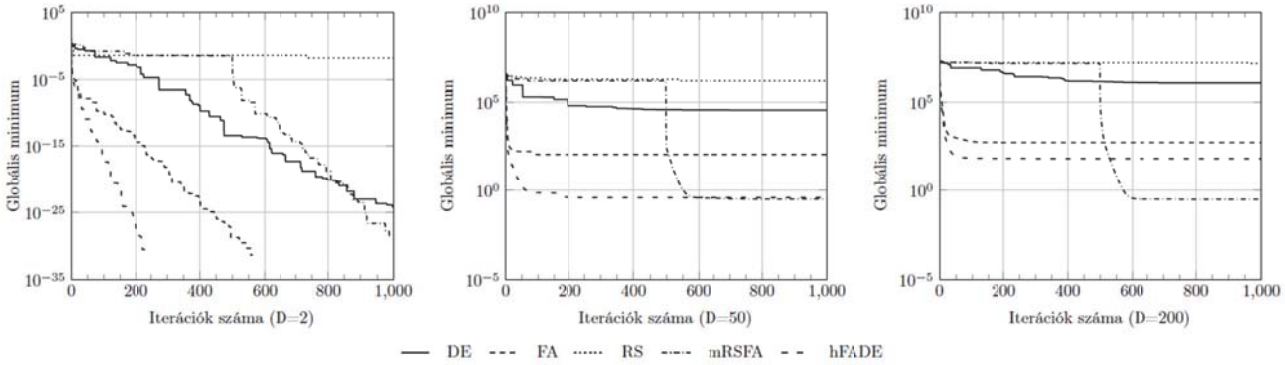
2. ábra Konvergencia Griewank függvény esetén



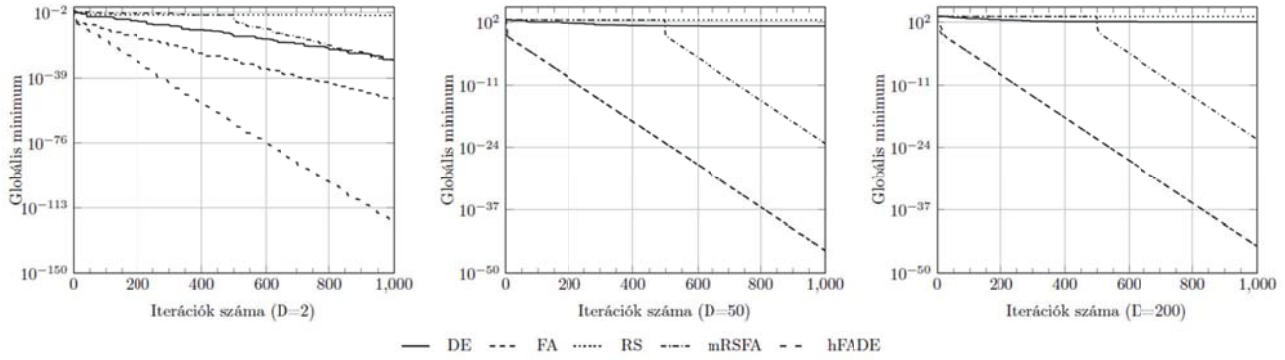
3. ábra Konvergencia Levy függvény esetén



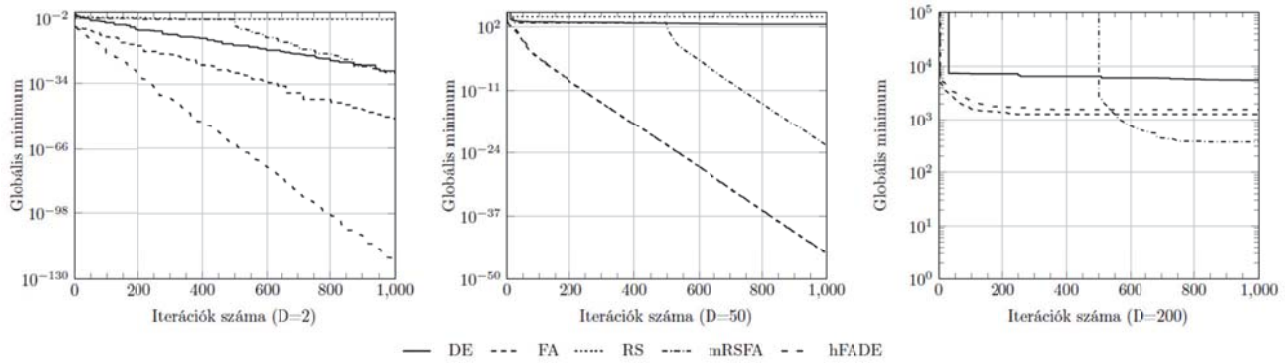
4. ábra Konvergencia Rastrigin függvény esetén



5. ábra Konvergencia Rosenbrock függvény esetén



6. ábra Konvergencia Sphere függvény esetén



7. ábra Konvergencia Zakharov függvény esetén

7. ÖSSZEFOGLALÁS

A cikkben bemutatásra kerültek alap és ezekből kombinált evolúciós módszerek, melyek különböző teszt függvényekkel lettek vizsgálva. Ezek eredményét az 1 - 7 ábrák szemléltetik. Balról jobbra a dimenziók száma $D = 2$, $D = 50$ és $D = 200$. Kis változójú problémáknál mind az eredeti módszerek, mind a többszintű és hibrid módszerek gyorsan tartanak az optimum felé. A változók számának növekedésével ez a képesség egyre jobban romlik, és egyre nagyobb valószínűséggel csak lokális minimumot talál. Ez olyannyira megfigyelhető, hogy a $D = 200$-as problémáknál közel biztosan kijelenthető, hogy lokális minimumot talál, ha létezik optimum.

Az összetett eljárások a jelen teszt függvényekkel végzett szimulációkban hatékonyabbak voltak az alap algoritmusoknál. Sajnos mivel az evolúciós módszerek sosem száz százalékban determinisztikusak, mint a gradiens alapúak, ezért teljes biztonsággal nem állítható, hogy minden létező problémánál hatékonyabbak lesznek. Csak a jelen teszt körülmények között voltak eredményesebbek és nagy rá a valószínűség, hogy a mérnöki problémák megoldása során is azok lesznek.

8. 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".

9. IRODALOM

[1] SIMON, D.: Evolutionary Optimization Algorithms, 2013

[2] LIU Y., PASSINO K. M.: Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviours, *Journal of Optimization Theory and Applications*, (2002), pp. 603-628.

[3] COLORNI A., DORIGO M., M. V., Distributed Optimization by Ant Colonies, 1991.

[4] KARABOGA D., BASTURK B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimizatio*n", (2007), pp. 459-471.

[5] KIRAN M., BABALIK A.: Improved Artificial Bee Colony Algorithm for Continuous Optimization Problems, *Journal of Computer and Communications*, (2014), pp. 108-116.

[6] KARABOGA D., AKAY B.: A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, *Applied Soft Computing*, (2011), pp. 3021-3031.

[7] KENNEDY J., EBERHART R.: Particle swarm optimization, Neural Networks, 1995. Proceedings., I*EEE International Conference on*, (1995), pp. 1942-1948.

[8] YANG XIN-SHE: A New Metaheuristic Bat-Inspired Algorithm, *Nature Inspired Cooperative Strategies for Optimization* (NICSO 2010), (2010)

[9] YANG X. S., SUASH D.: Cuckoo Search via Levy flights, *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, (2009), pp. 210-214.

[10] REYNOLDS R. G.: An introduction to cultural algorithms, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, (1994), pp. 131-139.

[11] STORN R., PRICE K.: Differential Evolution -- A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, *Journal of Global Optimization*, (1997), pp. 341-359.

[12] ZHENYU Y., KE T., XIN Y.: Self-adaptive differential evolution with neighbourhood search, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, (2008), pp. 1110-1116.

[13] BREST J., MAUVEC M. S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies, *Soft Computing*, (2011), pp. 2157-2174.

[14] BREST J., ZAMUDA A., B. B., M. S. M., V. Z.: High-dimensional real-parameter optimization using Self-Adaptive Differential Evolution algorithm with population size reduction, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, (2008), pp. 2032-2039.

[15] XIN-SHE YANG: Firefly Algorithms for Multimodal Optimization, *Stochastic Algorithms: Foundations and Applications: 5th International Symposium,* (2009), pp. 169-178.

[16] IZTOK F., XIN-SHE Y., JANEZ B., IZTOK F.: Memetic Self-Adaptive Firefly Algorithm, *Swarm Intelligence and Bio-Inspired Computation*, (2013), pp. 73-102.

[17] ZHANG L., LIU L., XIN-SHE Y., DAI Y.: A Novel Hybrid Firefly Algorithm for Global Optimization, *PLOS ONE*, (2016), pp. 1-17.

[18] KOTA L., JÁRMAI K.: Application of Multilevel Optimization Algorithms, *Advances in Structural and Multidisciplinary Optimization: Proceedings of the 12th World Congress of Structural and Multidisciplinary Optimization (WCSMO12)*, (2018), pp. 710-715.