FUTÓDARUHÍD OPTIMÁLÁSA PÁRHUZAMOS FPA ALGORITMUSSAL GPU-N

OPTIMIZATION OF OVERHEAD TRAVELLING CRANE WITH FPA ALGORITHM ON GPU

Nagy Szilárd[*], *Jármai Károly*[**]

ABSTRACT

Nature-inspired evolutionary optimization algorithms are powerful tools for solving non-linear problems. Sometimes they require huge computation capability, and they may be slow. In this paper, we propose a possible parallelization method for computation of base FPA algorithm and one group of fitness function. Proposed method simulated with three test function and an optimization of main girder of overhead travelling crane.

1. BEVEZETÉS

A meta-heurisztikus és evolúciós módszerek napjaink hatékony eszközei nem lineáris folytonos, vagy korlátos optimalizálási problémák megoldására. Ezeket az alábbi formában lehet összefoglalni:

$$\begin{aligned} &min \quad f(x) \\ &ha \quad g_i(x) \le 0 \quad i = 1, \cdots, q \\ &\quad\;\; h_j(x) = 0 \quad j = q+1, \cdots, n \\ &x = (x_1, x_2, \cdots, x_D) \in F \subseteq S \end{aligned} \tag{1}$$

ahol F a lehetséges megoldások halmaza és S pedig a keresési tér. Az evolúciós algoritmusok felépítése és működése hasonló. Ez azt jelenti, hogy a kezdeti populáció egyedeit módosítják természet inspirálta technikák segítségével. A módosított egyedekkel pedig minden iterációs lépésben kiszámolják a célfüggvényt. Egyszerűbb algoritmusok esetében – úgy mint a differenciál evolúció [1], részecskeraj optimalizáció [2], vagy virág beporzási algoritmus (FPA) [3], stb.:

$$FE = n_{pop} * n_{iter} \tag{2}$$

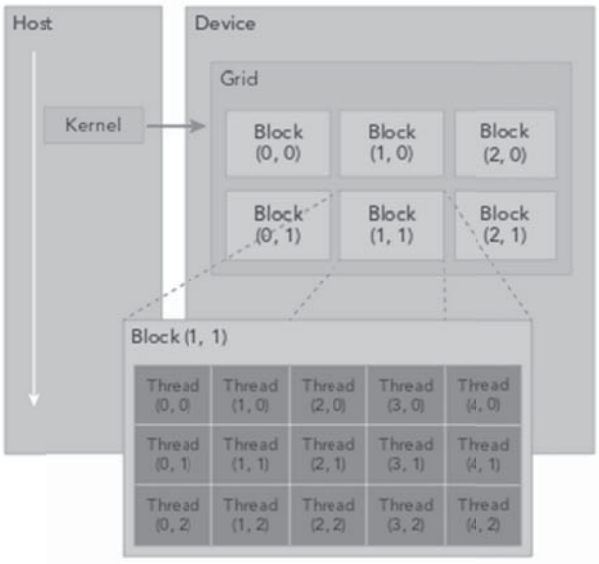
Belátható, hogy néhány esetben komoly erőforrás és idő igénye lehet egy-egy optimalizációnak.

Párhuzamosítással csökkenthető a szükséges számítási idő. Az evolúciós algoritmusok különböző módokon párhuzamosíthatóak [4]:

- globális modell: legegyszerűbb módszer csak az elemi műveletek futnak párhuzamosan
- regionális modell: a teljes populáció azonos méretű kisebb csoportokra kerül felbontásra, és ezeken a populációkon történik egymástól függetlenül, párhuzamosan az optimalizáció. Meghatározott időközönként a populációk között a kommunikációt a migráció biztosítja.
- lokális modell: minden egyed egy külön neki dedikált szálon, mikroprocesszoron fut és csak a szomszédaival kommunikál.

Jelen cikkben bemutatunk egy lehetséges módszert az FPA algoritmus párhuzamosítására, és javaslunk egy módszert a célfüggvény párhuzamos számítására is.

2. CUDA KÖRNYEZET



1. ábra CUDA felépítése [10]

Ma már a GPU-k (Graphical Processor Unit) nem csak a grafikus megjelenítést és az azokhoz szorosan kapcsolódó számítások elvégzését támogatják. Olcsó és hatékony eszközei az általános célú, tudományos számításoknak. Sikeresen kiaknázzák az általuk nyújtott lehetőségeket a topológiai optimálásban [5, 6], szerkezet

* *PhD hallgató, Miskolci Egyetem, e-mail: nagysz@gmail.com*
** *egyetemi tanár, Miskolci Egyetem, e-mail: jarmai@uni-miskolc.hu*

optimálásban [7] és gyártási folyamatok szimulálásában [8, 9]. SIMD (egyazon művelet, több adaton) programozási modell implementálásával, egy masszívan párhuzamos környezetet kínálnak. Napjainkban két keretrendszert alkalmaznak ezen kártyák programozására. Az első az OpenCL, ami egy nyílt forráskódú rendszer. A második pedig csak kizárólag az NVIDIA kártyákat támogató CUDA.

CUDA a C/C++ nyelv egy kiegészítése, amit formálisan szoktak CUDA-C-nek is hívni. Sok egyéb kiegészítés mellett a legfontosabb, hogy három függvény típussal egészíti ki a C nyelvet:

- „host" függvények: számítógép processzorán futnak és csak is innen hívhatóak. Egy az egyben megegyeznek az eredeti C/C++ függvényekkel.
- „kernel" függvények: a CPU által hívott függvények de GPU-n futnak, több szálon összhangban SIMD modell jellemzőivel. **_global_** előtaggal lehet őket deklarálni.
- „device" függvények: csak kernel függvényből hívható függvények és grafikus kártyán futnak. **_device_** előtaggal lehet őket deklarálni.

Az előző felsorolás is említi, hogy a kernel függvények azok, amik több szálon futnak a grafikus kártyán. Ezek a szálak, blokkokba vannak szervezve. A blokkok pedig rácsba, lásd 1. ábra. A rácsok és blokkok háromdimenziós koordinátákkal jellemezhetőek. Ezek mérete erősen függ az alkalmazott hardver típusától és annak paramétereitől. A blokkok és rácsok aktuális mérete, pedig a futtatandó probléma méretétől és komplexitásától függ. Minden koordináta változó egyértelmű azonosításához ad a CUDA egy-egy rendszer változót. CUDA rendszer részletesebb bemutatása, és leírása megtalálható [10] és [11] irodalmakban.

3. PÁRHUZAMOS VIRÁG BEPORZÁSI ALGORITMUS (FPA)

A növények fő reprodukciós folyamata a virág beporzás. Két jellemző mechanizmusa van az abiotikus (helyi) és biotikus (globális). Pollenek hosszú utat tehetnek meg különböző beporzók segítségével – madarak, rovarok, szél stb. A helyi beporzáshoz nincs szükség beporzók segítségére. Ezt a modellt adoptálja az FPA algoritmus [3].

A biotikus (globális) mechanizmust az alábbi matematikai formula modellezi

$$\overline{x}_i^{(G+1)} = \overline{x}_i^{(G)} + L\left(\overline{x}_i^{(G)} - \overline{g}_*^{(G)}\right) \tag{3}$$

ahol $\bar{x}_i$ az i. egyed (pollen), L Levy eloszlású véletlen szám, és $\overline{g}_*$ az aktuálisan megtalált minimum helye. A lokális beporzás modellje pedig

$$\overline{x}_i^{(G+1)} = \overline{x}_i^{(G)} + \epsilon\left(\overline{x}_j^{(G)} - \overline{x}_k^{(G)}\right)$$
$$i \neq j \neq k \in [1, D] \tag{4}$$

ahol $\epsilon \in [0,1]$ egyenletes eloszlású véletlen szám és i, j, k változók pedig független indexek. Egy p valószínűség függvényében kerül kiválasztásra ez egyik illetve a másik módszer. Az algoritmus részletesebb bemutatása megtalálható a [3]-ban.

A (3)-(4) egyenletekben látható konstansok legyenek tárolva egy-egy n_p nagyságú vektorban $(\bar{p}, \bar{L}, \bar{\epsilon}, \bar{j}, \bar{k})$. Ezek a vektorok egy-egy véletlen számot tárolnak, amik egymástól függetlenül több párhuzamos szálban generálhatóak a *cudaRAND* függvény könyvtár segítségével. A populáció pedig a G. iterációs lépésben pedig legyen tárolva egy $\bar{\bar{P}}$ mátrixban az alábbiak szerint

$$\bar{\bar{P}}^{(G)} = \begin{bmatrix} \bar{x}_1^{(G)} \\ \bar{x}_2^{(G)} \\ \vdots \\ \bar{x}_r^{(G)} \\ \vdots \\ \bar{x}_{n_p}^{(G)} \end{bmatrix} = \begin{bmatrix} \bar{x}_{1,1}^{(G)} & \bar{x}_{1,2}^{(G)} & \cdots & \bar{x}_{1,s}^{(G)} & \cdots & \bar{x}_{1,D}^{(G)} \\ \bar{x}_{2,1}^{(G)} & \bar{x}_{2,2}^{(G)} & \cdots & \bar{x}_{2,s}^{(G)} & \cdots & \bar{x}_{2,D}^{(G)} \\ \vdots & \vdots & & \vdots & & \vdots \\ \bar{x}_{r,1}^{(G)} & \bar{x}_{r,2}^{(G)} & \cdots & \bar{x}_{r,s}^{(G)} & \cdots & \bar{x}_{r,D}^{(G)} \\ \vdots & \vdots & & \vdots & & \vdots \\ \bar{x}_{n_p,1}^{(G)} & \bar{x}_{n_p,2}^{(G)} & \cdots & \bar{x}_{n_p,s}^{(G)} & \cdots & \bar{x}_{n_p,D}^{(G)} \end{bmatrix} \tag{5}$$

könnyen belátható, hogy ebben a formában egy ideiglenes $\bar{\bar{P}}^{(t)}$ populáció mátrix minden egyes eleme egymástól függetlenül számítható párhuzamosan követve SIMD modellt

$$P_{r,s}^{(t)} = \begin{cases} P_{r,s}^{(G)} + L_r\left(P_{r,s}^{(G)} - g_{*,s}^{(G)}\right) & p_r \leq rand(0,1) \\ P_{r,s}^{(G)} + \epsilon_r\left(P_{j_r,s}^{(G)} - P_{k_r,s}^{(G)}\right) & egyébként \end{cases} \tag{6}$$

a szelekció során pedig párhuzamosan kiválasztható a jobb egyed

$$P_{r,s}^{(G+1)} = \begin{cases} P_{r,s}^{(t)} & ha\ \mathcal{F}\left(\overline{P}_r^{(t)}\right) \leq \mathcal{F}\left(\overline{P}_r^{(G)}\right) \\ P_{r,s}^{(G)} & egyébként \end{cases} \tag{7}$$

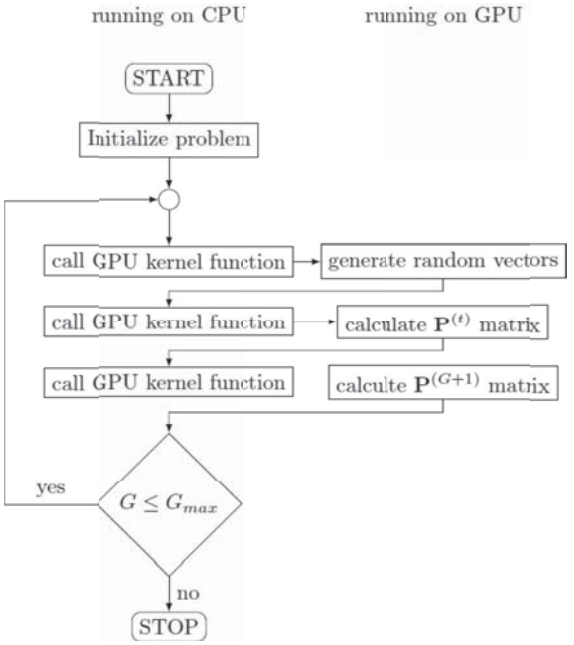
ahol $\mathcal{F}()$ a fitnesz függvény.

A javasolt módszer folyamatábráját szemlélteti a 2 ábra. Látható, hogy a fő lépések egymás után szekvenciálisan hajtódnak végre, de a számításigényes műveletek párhuzamosan futnak a grafikuskártyán.

4. FITNESSZ FÜGGVÉNY PÁRHUZAMOS SZÁMÍTÁSA MÓDOSÍTOTT PÁRHUZAMOS REDUKCIÓVAL

Evolúciós algoritmusok alkalmazása során a fitnesz függvény egy speciális függvény, mely büntetőfüggvény kialakítású. Ez rangsorolja iterációs

lépésenként az egyedeket. A felépítésük algoritmusonként és megoldandó feladatonként különböző lehet. A különbözőség ellenére van egy közös tulajdonságuk, mindegyik egy redukciót hajt végre. Azt jelenti, hogy a több dimenziós térhez egyetlen számot rendel. Mérnöki problémák esetén ez az esetek többségében egy valós szám.



2. ábra Javasolt párhuzamos FPA algoritmus folyamatábrája

$$\mathcal{F} = \mathcal{F}\left(f(\bar{x}), g_1(\bar{x}) \cdots g_q(\bar{x}), h_1(\bar{x}) \cdots h_r(\bar{x})\right) \quad (8)$$

$$\mathcal{F}: S^D \mapsto \mathbb{R} \quad (9)$$

ahol S^D a D dimenziójú keresési tér.

A párhuzamos redukció egy jól ismert eleme a párhuzamos technikák eszköztárának.

1. definíció [12]**:** legyen $\mathcal{B}$ alaphalmaz és legyen $\otimes$ asszociatív bináris operátor értelmezve a $\mathcal{B}$ halmaz elemein. Továbbá legyen kiszámolható egyetlen lépésben és zárt $\mathcal{B}$-re nézve. Ha

$$\bar{x} = \langle x_1, x_2, \cdots, x_p \rangle \in \mathcal{B} \quad (10)$$

akkor a párhuzamos redukció

$$\langle x_1, x_1 \otimes x_2, \cdots, x_1 \otimes x_2 \otimes \cdots \otimes x_p \rangle \quad (11)$$

Az algoritmus implementációja, és futásteljesítményre történő optimálása több irodalomban is megjelent [13,14]. Sajnos így ebben a formában egy az egyben nem lehet alkalmazni a fitnesz függvény kiszámítására, mert ritkán tartalmaz csak asszociatív műveleteket és ritkán számolható ki egyetlen egy lépésben.

Az eredeti fitnesz függvény felbontható egyszerűbb al-függvények variációjára

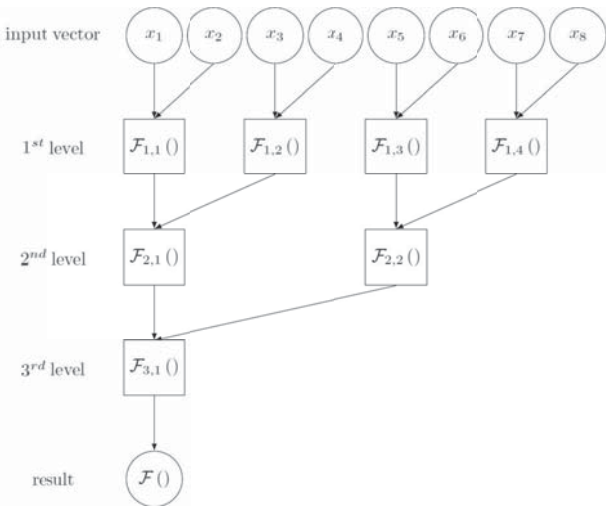
$$\mathcal{F}(x) = \mathcal{F}_1(\bar{x}_{v1}) \otimes_1 \mathcal{F}_2(\bar{x}_{v2}) \otimes_2 \cdots \otimes_{n-1} \mathcal{F} \quad (\bar{x}_{vn}) \quad (12)$$

ahol az $\bar{x}_{v1}, \bar{x}_{v2} \cdots \bar{x}_{vn}$ vektorok elemei az eredeti $\bar{x}$ vektor elemeinek a variációja. Az így kapott al-függvények további egyszerűbb függvényre felbonthatók. Azt a felbontást addig kell ismételni míg az eredeti függvény olyan egyszerű függvényekből nem áll, amire igaz, hogy két változótól és egy α konstanstól függ és kiszámíthatóak egy lépésben.

$$\mathcal{F} = \mathcal{F}\left(x_i, x_j, \alpha\right) \quad (13)$$

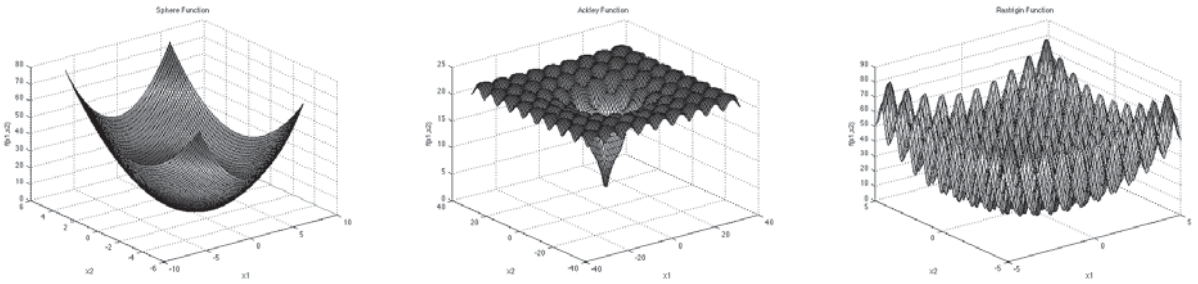
ahol i, j független indexek.

Az alapvető matematikai műveleteket, mint az összeadást, kivonást, szorzást, osztást stb. programozás technikailag függvényként szükséges kezelni. Az al-függvények hierarchiája ábrázolható egy fa szerkezetben, mint azt a 3. ábra szemlélteti. A fa levelei – a bemeneti változók – az eredeti $\bar{x}$ vektor elemeinek a variációja. A csomópontok tartalmazzák a felbontásból származó „egyszerű" függvényeket. Az alsóbb szintek bemenő adatai pedig minden esetben az őt megelőző szinten elhelyezkedő függvények eredményei. Látható, hogy egy adott szinten található függvények nem függenek egymástól ezért kiszámíthatóak párhuzamosan.



3. ábra Függvények hierarchiája fa szerkezetben

Az eredeti párhuzamos redukció algoritmusát, pedig úgy kell módosítani, hogy a szokásos bináris művelet helyett a korábban megalkotott függvényhívás legyen az azonos művelet minden egyes szálban. Ez C/C++ és CUDA környezetben könnyen megtehető mivel mind a két rendszer támogatja a mutató alapú függvény hívást.



a) Sphere függvény *b) Ackley's függvény* *c) Rastrigin függvény*

4. ábra Függvények hierarchiája fa szerkezetben

5. SZIMULÁCIÓ

Jelen cikkben az előzőekben vázolt módszerrel iterációs lépésenként elérhető átlagos sebesség növekedést vizsgáltuk, amit az alábbi formában értelmeztünk:

$$t = \frac{\frac{1}{n}\sum_{i=1}^{n} t_i^{(seq)}}{\frac{1}{m}\sum_{j=1}^{m} t_j^{(par)}} \tag{14}$$

ahol $t_i^{(seq)}$ a szekvenciális futáshoz szükséges idő, $t_j^{(par)}$ a párhuzamos futáshoz szükséges idő és m, n a vizsgált minták száma.

A szimulációt három, az evolúciós algoritmus vizsgálatához gyakran használt tesztfüggvénnyel és egy valós tervezési feladattal végeztük el.

A három tesztfüggvény az alábbiak:

- Sphere függvény (4.a ábra):

$$\mathcal{F}(\bar{x}) = \sum_{i=1}^{D} x_i^2 \tag{15}$$

- Ackley's függvény (4.b ábra):

$$\mathcal{F}(\bar{x}) = -a\exp\left(-b\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos cx_i\right) + a + \exp(1) \tag{16}$$

ahol $a = 20$, $b = 0.2$ és $c = 2\pi$

- Rastrigin függvény (4.c ábra):

$$\mathcal{F}(\bar{x}) = 10D + \sum_{i=1}^{D}\left(x_i^2 - 10\cos 2\pi x_i\right) \tag{17}$$

Ezek a problémák folytonos optimalizálási feladatok.

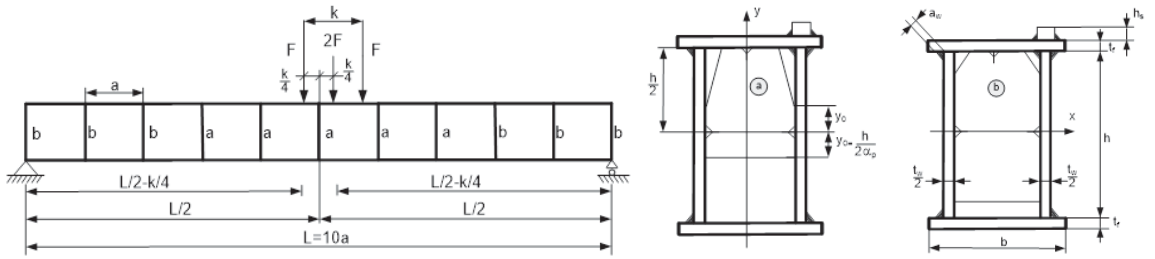
A valós tervezési feladat pedig egy híddaru szekrény szekrényszelvényű fő tartójának az optimalizálása. Egy egyfeltételes, korlátos optimalizálási probléma.

$$\mathcal{F}(\bar{x}) = f(\bar{x}) + \sum_{i=1}^{n} p(g_i(\bar{x})) \tag{18}$$

ahol n a korlátok száma $p()$ pedig az alábbiak szerint definiált büntető függvény:

$$p(\bar{x}) = \begin{cases} 0 & \text{ha } g_i(\bar{x}) \le 0 \\ \infty & \text{egyébként} \end{cases} \tag{19}$$

Az ismeretlenek: a tervezési változók $\bar{x}^T =$



5. ábra Szekrény szelvényű főtartó struktúrája, szekrény szelvény jellemző méretei, és diafragmák

Úgy, mint gerinclemez magassága h, vastagságának duplája t_w, övlemez szélessége b és vastagsága t_f lásd 5. ábra. A teljes tervezési és optimalizálási folyamat részletes leírása bele értve tervezési követelményeket is megtalálható [14],[15] irodalmakban.

Az optimálás célfüggvénye a költség, mely az anyag, hegesztési és utómunkálási költségeket veszi figyelembe.

- anyagköltség:

$$K_m = \hat{K}_{m1} h t_w + \hat{K}_{m2} b t_f + \hat{K}_{m3} b h \tag{20}$$

- felső övlemez összefűzési és hegesztési költsége

$$K_{w1} = \hat{K}_{w1,1} \sqrt{\hat{K}_{w1,2} h t_w + \hat{K}_{w1,3} b t_f + \hat{K}_{w1,4} b h} + \hat{K}_{w1,5} \tag{21}$$

- sín összeállítási hegesztési költsége

$$K_{w11} = \hat{K}_{w11,1} t_w^{1,94} \tag{22}$$

- diafragmák összeállítási és hegesztési költsége

$$K_{w12} = \hat{K}_{w12,1} b t_w^2 + \hat{K}_{w12,1} h t_w^2 \tag{23}$$

- alsó övlemez összefűzési és hegesztési költsége

$$K_{w2} = \hat{K}_{w2,1} \sqrt{\hat{K}_{w2,2} h t_w + \hat{K}_{w2,3} b t_f + \hat{K}_{w2,4} b h} + \hat{K}_{w2,5} \tag{24}$$

- a két gerinc összeállítási és hegesztési költsége 11db 1500mm-es lemezből

$$K_{w3} = \hat{K}_{w3,1} \sqrt{h t_w} + \hat{K}_{w3,1} h t_w^{1,94} \tag{25}$$

- két övlemez lemez összeállítási és hegesztési költsége 11db 1500mm-es lemezből

$$K_{w4} = \hat{K}_{w3,1} \sqrt{b t_f} + \hat{K}_{w3,1} b t_f^{1,94} \tag{26}$$

- utókezelés költsége

$$K_t = \hat{K}_{t,1} b \tag{27}$$

- teljes költség

$$f(\bar{x}) = K_m + K_{w1} + K_{w11} + K_{w12} + K_{w2} + 2K_{w3} + 2K_{w4} + K_t \tag{28}$$

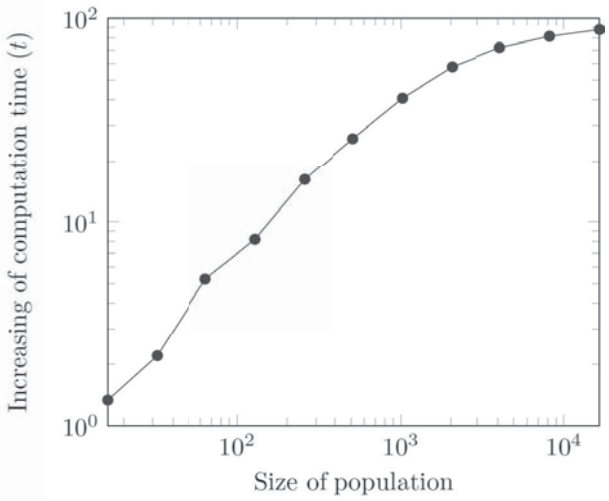
ahol $\hat{}$ mennyiségek konstansok, amik levezethetőek a [14] és [15] irodalmak egyenleteiből. A (20)-(28) egyenletek felépítése megegyezik a 4. fejezetben párhuzamos számításhoz javasolt függvény felépítéssel.

A korlátok pedig a tervezési feltételekből következnek úgymint szilárdságtani feltételek, határkarcsúság, fáradási feltétel és lehajlási feltétel. A példa kedvéért itt csak a hajlításból adódó feltételt mutatjuk be. A többi képlet a már említett [14] és [15] egyenleteiből hasonlóan levezethető és felépíthető a szükséges fa szerkezet, hasonlóan, mint a költség számítás esetében.

$$\sigma_{hx} = \hat{\sigma}_{hx,1} \frac{t_w}{h t_w + 6 b t_f} + \hat{\sigma}_{hx,2} \frac{b t_f}{h\left(h t_w + 6 b t_f\right)} \tag{29}$$

$$\sigma_{hy} = \hat{\sigma}_{hy,1} \frac{h t_w}{b\left(3 h t_w + 2 b t_f\right)} + \hat{\sigma}_{hy,2} \frac{t_f}{2 b t_f + 3 h t_w} \tag{30}$$

$$g_1 = -\sqrt{\hat{g}_{1,2} - \hat{g}_{1,3} \frac{\sigma_{hx} + \sigma_{hy}}{\sigma_{hx} - \sigma_{hy}} + \hat{g}_{1,4} \frac{\left(\sigma_{hx} + \sigma_{hy}\right)^2}{\left(\sigma_{hx} - \sigma_{hy}\right)^2}} + \hat{g}_{1,1} \tag{31}$$

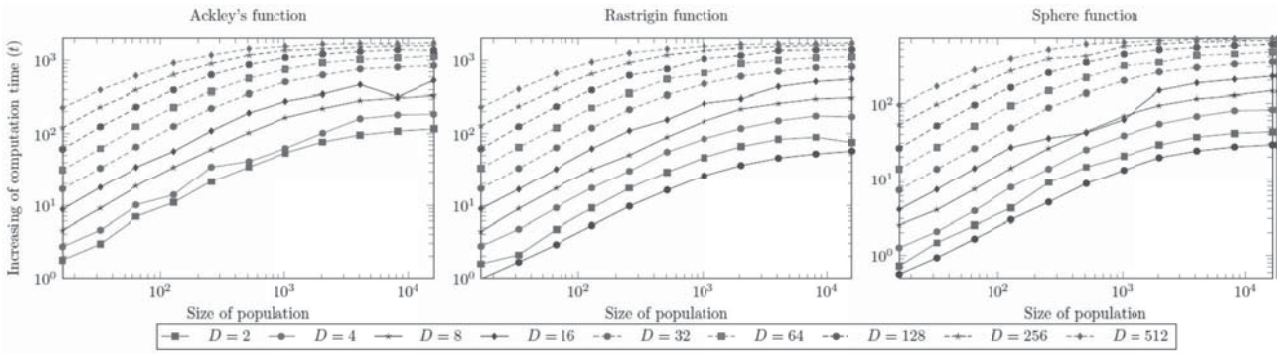


6. ábra Futásidő változása a populációméret függvényében futódaru optimálása során

6. KONKLUZIÓ

A három tesztfüggvény esetén a szimuláció több különböző populáció mérettel és változószámmal lett elvégezve. A változók száma 2 és 1024 között változott, míg a populáció méret 16 és 16384 között. Az elérhető sebesség növekedés erősen függ probléma nagyságától, mint ahogyan látható a 7. ábrán is. Kis populáció mérettel és kevés változóval alig érhető el nagy sebesség növekedés és semmi sem indokolja a párhuzamosítást. Ellenben, ha nagyméretű a megoldandó probléma megközelítőleg ezerszeres sebesség növekedés is elérhető.

A futódaru híd optimálása során nyerhető sebesség növekedés hasonlóan alakul a tesztfüggvényekhez (6. ábra). Itt csak a populáció méretét lehetett növelni, mivel az ismeretlen változók száma adott.



7. ábra Futásidő változása a populációméret függvényében a teszt függvények optimálása során

7. 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.

8. IRODALOM

[1] STORN R., PRICE K.: Differential evolution – a simple and efficient heuristic for global optimization over continous spaces, *Jurnal of Global Optimization*, Vol.: 11, No.: 4, (1997), pp. 341-359, doi:10.1023/A:1008202821328

[2] XIE X.F., ZHANG W.J., YANG Z.L.: Adaptive particle swarm optimization on individual level, *6th International Conference on Signal Processing*, Vol.: 2, (2002), pp. 1215-1218, doi: 10.1109/ICOSP.2002.1180009

[3] YANG X.S.: Flower pollination algorithm for global optimization, *Unconventional and Natural Computation*, Berlin, (2012) pp. 240-249

[4] BURGOLYA I.: *Optimalizálás evolúciós számításokkal*, Typotex kiadó, Budapest, 2012

[5] DURATE L.S., WALDEMAR C., ANDERSON P., IVAN I.F.: Glaucio, P.: Polytop++: an efficient alternative for serial and prallel topology optimization on cpus & gpus, *Structural and Multidisciplinary Optimization*, Vol.:52, No.:5, (2015), pp 845-859 doi:10.1007/s00158-015-1252-x

[6] XIA Z., WANG Y., WANG Q., MEI C.: GPU parallel strategy for parameterized lsm-based topology optimization using isogemetric analysis, *Structural and Multidisciplinary Optimization*, Vol.:56, No.:2, (2017), pp. 413-434 doi: 10.1007/s00158-017-1672-x

[7] KALAVARAPU V., WINER E.: A study of graphics hardware accelerated particle swarm optimization with digital pheromones, *Structural and Multidisciplinary Optimization*, Vol: 51, No.: 6, (2015), pp. 1281-1304, doi: 10.1007/s00158-014-1215-7

[8] ROTHLIN M., KLIPPEL H.: AFRASIABI M. WEGENER K.: Metal cutting simulation using smoothed paricle hydrodynamics on the gpu, *Structural and Multidisciplinary Optimization*, Vol.: 102, No.:9, (2019), pp 3445-3457, doi: 10.1007/s00170-019-03410-0

[9] WANG J., ZHANG D., LOU M., ZHANG Y.: A gpu-based tool parameters optimization and tool orientation control method for four-axis milling with ball-end cutter, *The International Jurnal of Advanced Manufacturing Technology*, Vol.: 102, No.: 5, pp. 1107-1125, doi: 10.1007/s00170-018-2954-1

[10] CHENG J., GROSSMAN M., MCKERCHER T.: *Professional CUDA C programming*, John Wiley and Sons Inc., Indianapolis, 2014

[11] SANDERS J., KANDROT E.: *CUDA by example – An introduction to general purpose GPU programming*, Addison-Wesley, Boston, 2011

[12] IVÁNYI A.: *Párhuzamos algoritmusok*, ELTE IK, Budapest, 2010

[13] MARTÍN P.J., AYUSO L.F., TORRES R., GAVILANES A.: Algorithmic strategies for optimizing the parallel reduction primitive in cuda, *International conference on High Performance Computing Simulation*, (2012), pp. 511-519

[14] NAGY SZ., JÁRMAI K.: Application of the firefly algorithm for the optimization of cranes. Advances and Trends in Engineering Sciences and Technologies III: Proceedings of the 3rd *International Conference on Engineering Sciences and Technologies (ESaT 2018)*, Editors: Mohamad Al Ali, Peter Platko, pp. 203-210. CRC Press, ISBN 9780367075095

[15] JÁRMAI K., FARKAS J.: *Fémszerkezetek innovatív tervezése*, Gazdász-Elasztik Kiadó és Nyomda, Miskolc, (2015)