Miskolci Egyetem
Gépészmérnöki és Informatikai Kar

evolúciós módszerek alkalmazása a szerkezetoptimálásban

PhD értekezés

Készítette:

Nagy Szilárd
villamosmérnök (BSc)
okleveles gépészmérnök

Sályi István Gépészeti Tudományok Doktori Iskola
Gépek és Szerkezetek Tervezése Tématerület
Mérnöki Szerkezetek Tervezése Témacsoport

Doktori Iskola vezető:
Vadászné Prof. Dr. Bognár Gabriella
a műszaki tudomány doktora, egyetemi tanár

Témacsoport vezető:
Prof. Dr. Jármai Károly
egyetemi tanár

Tudományos vezetők:
Prof. Dr. Jármai Károly
egyetemi tanár
Dr. Baksa Attila
egyetemi docens

Miskolc
2022

Tartalomjegyzék

Tudományos vezetői ajánlás

Nagy Szilárd a 2017/2018. tanév II. félévében kezdte PhD doktori tanulmányait a Sályi István Doktori Iskolában. Már a felvételit megelőzően bemutatott kutatási tevékenysége, valamint a témaválasztása is sikeres, eredményes doktori képzés lehetőségét vetítette előre.

Nagy Szilárd doktori képzési munkáját az Emerson Automation FCP Kft., Emerson FLMCP csoportnál végzett mérnöki tevékenysége mellett folytatta. A képzési követelményekben előírt tantárgyi kötelezettségeinek folyamatosan magas színvonalon tett eleget. A képzése és kutatási tevékenysége szempontjából fontos tantárgyakat jól strukturált összetételben, kiváló eredménnyel teljesítette.

Szinte a doktori képzés megkezdésétől kezdve oktatási tevékenységet is folytatott és emellett kiemelkedő kutatási tevékenységet fejtett ki. Kezdetben a kutatómunkája a különböző fémszerkezetek tervezése, és az optimáló algoritmusok vizsgálatára irányult, amely hamarosan a grafikus kártya programozásával (GPU) bővült, legvégén pedig saját végeselemes számításokkal. Ezen a területen különösen kiemelendő az optimálás és a GPU programozás összekapcsolása.

Doktori kutatásaiból intenzív publikációs tevékenységet is folytatott. Kezdve a Doktori Fórumokon és Kutatószemináriumokon tartott előadásaival, számos hazai és nemzetközi konferencián vett részt és adott elő, valamint kutatási eredményeit több rangos szakcikkben is megjelentette.

Már a doktori képzési szakaszban végzett kutatási tevékenysége is jól alátámasztotta azt a sokoldalú és egyre elmélyültebb kutatótevékenységet, amely Nagy Szilárd egész doktori kutatási munkáját jellemezte. Az értekezés átgondolt, szisztematikus kutatási tevékenységet ismertet, magas szintű elméleti ismereteket és számítógépi szimulációkat és mind tartalmilag, mind kivitelét tekintve színvonalas tudományos munkát alkot.

PhD doktori kutatómunkáját nagyfokú önállósággal és precizitással végezte. Külön kiemelendő, hogy kiváló elméleti felkészültsége mellett, jó programozói érzékkel is rendelkezik: e két kiváló kutatói adottság egyrészt az értekezés elméleti fejtegetéseiben, másrészt az értekezés jelentős szimulációs részében is jól nyomon követhető.

Miskolc, 2022.06.02

.....	.....
Prof. Dr. Jármai Károly	Dr. Baksa Attila
témavezető	társ-témavezető

Előszó

A mai mérnöki feladatoknál egyre gyakoribb elvárás az optimális megoldás megtalálása, és az egyre jobban szigorodó energia-, költséghatékonysági elvárások mellett a lehető legmagasabb teljesítmény elérése. A feladatok egyre bonyolultabbak, melyeket hagyományos, analitikus úton már egyre nehezebb megoldani, vagy az alkotott modell nem elég pontos. Ilyen esetekben nyújthat újfajta eszközöket a mesterséges intelligencia, ugyanakkor a tudományág részeként említhetőek meg a sztochasztikus alapokon nyugvó evolúciós optimáló/kereső algoritmusok. Hagyományos deriváltakon alapuló módszerek helyett kínálnak alternatív hatékony eszközöket sok változós, nemlineáris függvények optimálására. Jelenleg ez egy intenzíven kutatott terület.

A disszertáció a bevezetést követően röviden áttekinti általánosságban az evolúciós algoritmusok felépítését, leggyakrabban alkalmazott technikáit. A gazdag és számos publikációval rendelkező irodalmi háttérre való tekintettel ez nem teljes, csak a legelterjedtebben alkalmazott, és jelen esetben is felhasználásra kerülő módszerekre/eljárásokra tér ki. A következőkben a több száz algoritmusból kiemelésre kerül néhány, és hatékonyságuk, teljesítőképességük alapján összehasonlításra kerülnek. A gyakorlatban is előforduló három feladat megoldásra kerül velük úgy, mint teherautó-plató kereszttartójának optimálása, futódaru főtartójának optimálása, és végül távvezeték torony alsó részének optimálása.

Dolgozatom egyik legértékesebb részének tekintem, mikor a 2.4. részben megfogalmazott feladatot példaként használva, a 4. fejezetben javaslatot teszek a masszívan párhuzamos környezetben történő optimálásra. A másik értékesnek tartott 2.5. részben pedig végeselem-módszert és az evolúciós technikát összekapcsolva, javaslatot teszek rácsos tartóval modellezhető szerkezet optimálási feladatának megoldására. A feladat megoldásához saját végeselem-módszeren alapuló rendszert fejlesztek ki. A javasolt módszerrel optimálom egy villanyoszlop alsó részét.

Alkalmazott jelölések

Alkalmazott matematikai jelölések	
$\lvert \cdot \rvert$	– halmazon értelmezve: a halmaz számossága, egyébként abszolút érték
$\lVert \cdot \rVert$	– 2-normán alapuló, euklideszi távolság
${}^{e}(\cdot)$	– bal felső indexben mennyiség: elem sorszáma
${}^{(G)}(\cdot)$	– bal felső indexben zárójelezett mennyiség: generáció
$(\cdot)^{*}$	– jobb felső indexben csillag: mutató jelölése
$(\cdot)'$	– jobb felső indexben vessző: elemhez kötött koordináta rendszerben értelmezett mennyiség
$d(\cdot, \cdot)$	– eukleidészi távolság
$O(\cdot)$	– algoritmus műveletigénye

Görög betűvel jelölt mennyiségek	
α_0	– utókezelést figyelembe vevő tényező
α_a	– alakpontatlansági tényező
α_{damp}	– véletlen mozgást csökkentő tényező
$\boldsymbol{\alpha}_l$	– véletlen mozgás konstansait tartalmazó vektor
β_g	– feszültség gradiens tényező
β_l	– vonzódási konstans
β_o	– oldalferdeség
γ_f	– biztonsági tényező anyagfáradáshoz
γ_l	– fényelnyelési tényező
γ_{M0}	– általános biztonsági tényező
$\Gamma(\lambda)$	– λ paraméterű gamma függvény
ϵ	– véletlen szám egyenletes eloszlás szerint
$\boldsymbol{\epsilon}$	– egyenletes eloszlás szerinti véletlen számokból épített vektor
ε	– folyáshatár-módosító tényező acélokra
${}^{e}\varepsilon$	– e elem fajlagos nyúlása
$\overline{\varepsilon}$	– eltérés, hibaérték
$\overline{\varepsilon}_{avg}$	– átlagos hiba
$\overline{\varepsilon}_{min}$	– legjobb hibaérték
$\varepsilon_{j,1}, \varepsilon_{j,2}, \ldots, \varepsilon_{j,n}$	– tartományhatárok egyenlőtlenségi feltételekhez
$\varepsilon_{k,1}, \varepsilon_{k,1}, \ldots, \varepsilon_{k,n}$	– tartományhatárok egyenlőségi feltételekhez
Θ	– bonyolultsági tényező
κ	– összefűzendő, összehegesztendő elemek száma

folytatás a következő oldalon

Görög betűvel jelölt mennyiségek	
$\overline{\lambda}$	– redukált karcsúsági tényező
Π_p	– teljes potenciális energia
$\delta\Pi$	– potenciális energia első variációja
ρ	– sűrűség
ρ_0	– fajlagos súly
ρ_{Al}	– alumínium sűrűsége
$\sigma_1,\ \sigma_2$	– hajlításból származó normál feszültség
σ_h	– normál feszültség
σ_y	– y tengellyel párhuzamos normál feszültség
σ_z	– z tengellyel párhuzamos normál feszültség
$\Delta\sigma_C$	– $2 \cdot 10^6$ ciklusszámhoz tartozó határfeszültség
$\Delta\sigma_D$	– $5 \cdot 10^6$ ciklusszámhoz tartozó határfeszültség
$\Delta\sigma_N$	– egy tetszőleges ciklusszámra átszámított határfeszültség
τ_t	– nyírófeszültség
$\tau_{xy},\ \tau_{yz}$	– nyírófeszültség
τ_V	– nyírófeszültség
$\Delta\tau_C$	– $2 \cdot 10^6$ ciklusszámhoz tartozó nyíró határfeszültség
$\Delta\tau_N$	– egy tetszőleges ciklusszámra átszámított határfeszültség
$\Phi(\ )_j$	– egyenlőségi feltétel megsértésének mértékével arányos függvény
$\Phi(\ )_k$	– egyenlőségi feltétel megsértésének mértékével arányos függvény
χ	– kifáradási tényező
Ψ_d	– dinamikus tényező
$\Psi_x,\ \Psi_y$	– feszültség arány

Latin betűvel jelölt mennyiségek	
a_2	– felső torony szélesség
a_w	– varrat jellemző mérete
A	– keresztmetszet területe
A_c	– kereszttartó keresztmetszet területe
A_p	– padlólemez együtt dolgozó lemezszélességgel vett keresztmetszet területe
A_V	– nyírási keresztmetszet
b	– övlemez szélessége
$\hat{C}_1, \hat{C}_2, \hat{C}_3, \ldots, \hat{C}_n$	– konstansok közelítő függvényekben
C_a	– gyártási időt befolyásoló konstans
$C_i,\ C_k$	– dinamikus büntetőfüggvény együtthatója
C_R	– keresztezési arány
$\mathbf{C}_{Rrec}$	– C_R keresztezési arányok vektora, mely sikeres visszahelyezést eredményeznek
C_{Rm}	– keresztezési arányok átlaga
C_w	– hegesztési időt befolyásoló konstans
$D_1, D_2, \ldots, D_n$	– CHS szelvény átmérői
E	– rugalmassági modulus
$\mathbf{f}$	– általánosított csomóponti terhelések vektora
$f(\cdot)$	– függvény általános jelölése

folytatás a következő oldalon

Latin betűvel jelölt mennyiségek

f_y	– folyáshatár
F	– erő általánosan
$F(\cdot)$	– tesztfüggvény
F_f	– felépítmény súlya
F_H	– vízszintes erő
$F_{n,0}$	– függvényérték eltolás
F_p	– „plató" maximális terhelhetősége
F_S	– skálázási tényező
F_V	– függőleges erő
F_y	– y tengellyel párhuzamos erő
$g_i(\cdot)$	– egyenlőtlenségi feltételt kifejező függvény
G	– generáció sorszáma
G_k	– „futómacska" súlya
G_t	– teljes generáció szám
h	– gerinclemez magassága
h_1	– rácsosztás magassága
h_2	– metszéspont magassága
h_a	– alsó toronyrész magassága a vizsgált síkban
h_c	– hosszmerevítő magassága
$h_k(\cdot)$	– egyenlőségi feltételt kifejező függvény
h_r	– osztásarány
h_s	– sín magassága
H_a	– alsó toronyrész magassága
H_f	– felső toronyrész magassága
$i,\ j$	– általános indexek
k	– kihajlási hossz tényező
k_k	– keréktávolság
k_m	– fajlagos anyagköltség
k_w	– fajlagos gyártási költség
$k_y,\ k_z,\ k_{\sigma z},\ k_\tau$	– biztonsági tényezők
$\tilde{k}_{\sigma x},\ \tilde{k}_{\sigma y},\ \tilde{k}_\tau,\ \tilde{k}_{\tau b}$	– helyi horpadási tényezők
K	– költség
$\mathbf{K}$	– merevségi mátrix
$\hat{\mathbf{K}}$	– egy keresztmetszet csoportba tartozó elemek merevségi mátrixa
$\hat{K}_1(\cdot), \ldots, \hat{K}_n(\cdot)$	– közelítő függvények
K_m	– anyagköltség
K_w	– gyártási költség
$\tilde{I}$	– szentjánosbogár fényessége
$I_x,\ I_y$	– másodrendű nyomatékok
L	– hosszúság
L_c	– kereszttartó segédvázon túlnyúló része
L_{ct}	– kereszttartó teljes hossza
L_p	– „plató" teljes hossza
L_w	– varrat hossz
m	– tömeg
m_c	– kereszttartók tömege

folytatás a következő oldalon

Latin betűvel jelölt mennyiségek

m_{opt}	– optimalizált tömeg
$m_{\%}$	– tömeg %-os változása
$\mathbf{M}$	– ortogonális forgatótenzor
M_h	– hajlító nyomaték
$M_{hx},\ M_{hy}$	– x ill. y tengely irányú hajlító nyomaték
n_c	– kereszttartók száma
n_{CR}	– $C_R$keresztezési arány frissítési intervalluma
n_D	– keresési tér dimenziója
n_e	– elemek száma
n_g	– megsértett egyenlőtlenségi feltételek száma
$n_{g,lim}$	– megsérthető egyenlőtlenségi feltételek száma
$n_{f1},\ n_{f2}$	– visszahelyezés során a $(G+1)$ generációba be nem került egyedek száma
n_{FE}	– függvényérték kiszámításának száma
n_h	– megsértett egyenlőségi feltételek száma
$n_{h,lim}$	– megsérthető egyenlőségi feltételek száma
n_{IT}	– iteráció szám
n_{lCR}	– C_R keresztezési arányhoz tartozó tanulási ciklus hossza
n_{lp}	– p_p valószínűségi változóhoz tartozó tanulási ciklus hossza
n_p	– populáció méret, populáció halmaz számossága
n_{racs}	– rácsosztások száma
$n_{s1},\ n_{s2}$	– visszahelyezés során $(G+1)$ generációba bekerült egyedek száma
N	– terhelési ciklusszám
${}^e\mathbf{N}$	– alakfüggvények mátrixa
${}^eN(\xi)$	– alakfüggvény
$\mathbf{o}$	– eltolás
$\mathbf{O}_p$	– pointer(ek) – mutató(k) – mátrixa
p	– megoszló terhelés
p_c	– egy kereszttartóra ható megoszló terhelés
$p_i(\cdot)$	– büntető függvény
p_p	– valószínűségi változó
$\mathbf{p}_p$	– valószínűségi változók vektora
p_r	– sín fajlagos önsúlya
p_s	– szekrényszelvény fajlagos önsúlya
P_h	– horog teher
Q	– nyíróerő
$r_1, r_2, r_3, \ldots, r_n$	– egymástól független véletlen indexek $r_1 \neq r_2 \neq r_3 \neq \ldots \neq r_n$
$\tilde{r}_i$	– büntető paraméter
$\tilde{R}$	– büntető konstans
R^2	– illeszkedési helyesség
s_3	– spektrumtényező
t	– lemez- / falvastagság
$\hat{t}$	– dimenziótlan sebesség növekedés
$t_1, t_2, t_3, \ldots, t_n$	– CHS szelvény falvastagsága
t_{alg}	– egy iterációs lépésen belül az algoritmushoz tartozó műveletek elvégzéséhez szükséges idő szekvenciálisan futtatva

folytatás a következő oldalon

Latin betűvel jelölt mennyiségek	
$\tilde{t}_{alg}$	– egy iterációs lépésen belül az algoritmushoz tartozó műveletek elvégzéséhez szükséges idő párhuzamosan futtatva
t_f	– övlemez vastagsága
t_{fgv}	– egy iterációs lépésen belül a fitnesz függvény kiszámításához szükséges idő szekvenciálisan futtatva
$\tilde{t}_{fgv}$	– egy iterációs lépésen belül a fitnesz függvény kiszámításához szükséges idő párhuzamosan futtatva
t_{it}	– egy teljes iterációs lépés kiszámításához szükséges idő szekvenciálisan futtatva
$\tilde{t}_{it}$	– egy teljes iterációs lépés kiszámításához szükséges idő párhuzamosan futtatva
t_p	– padlólemez vastagsága
t_s	– diafragma osztás
t_w	– gerinclemez vastagsága
$\mathbf{T}$	– transzformációs tenzor
T_i	– gyártási fázis ideje
T_w	– hegesztési idő
$\mathbf{u}$	– csomóponti elmozdulás vektor
$u_x,\ u_y,\ u_\xi$	– csomóponti elmozdulás x, y vagy ξ tengellyel párhuzamos komponense
$\tilde{\mathbf{u}}_i$	– próba vektor
U	– alakváltozási energia
$\mathbf{v}_i$	– i. mutáns vektor
V	– térfogat
w	– lehajlás
W	– külső erők munkája
$W_i,\ W_j$	– tényező
$W_x,\ W_y$	– keresztmetszeti tényezők
$\mathbf{x}$	– egyed
$\hat{\mathbf{x}}$	– eredeti egyed paramétereinek variációjából épített vektor
$\mathbf{x}_{best}$	– populáció legkisebb fitnesz értékű egyede
$\mathbf{x}_i$	– populáció i. egyede
$x_{i,j}$	– egyed j. tulajdonsága (paramétere)
$x_{L,i}$	– $\mathbf{x}$ vektor i. elemének alsó korlátja
$x_{U,i}$	– $\mathbf{x}$ vektor i. elemének felső korlátja
$\mathbf{X}$	– populáció egyedeiből épített mátrix
y_G	– súlyponti tengely magassága a szelvény szélétől mérve

Kalligrafikus betűvel jelölt mennyiségek	
$\tilde{\mathcal{F}}$	– egyszerű függvény konstansa
$\mathcal{F}(\ )$	– fitnesz függvény
$\Delta\mathcal{F}_{rec}$	– sikeres visszahelyezéssel elért fitneszérték változásból képzett vektor
$\hat{\mathcal{F}}(\ )$	– egyszerű függvény
$\mathcal{L}$	– Lévy eloszlás szerinti véletlen szám
$\mathcal{N}(m,\sigma)$	– művelet jelölése normál eloszlású, m középértékű, σ^2 szórású véletlen szám generálásához

folytatás a következő oldalon

Kalligrafikus betűvel jelölt mennyiségek	
$\mathcal{P}$	– populáció halmaza, röviden populáció
$\mathcal{P}_m$	– mutációval képzett egyedek halmaza
$\mathcal{P}_p$	– szülők halmaz, röviden szülők
$\mathcal{P}_r$	– rekombinációval képzett egyedek halmaza
$\mathcal{S}$	– keresési tér
$\mathcal{S}^D$	– D dimenziójú keresési tér
$\mathcal{U}(a,b)$	– művelet jelölése $[a,b)$ félig zárt intervallumon egyenletes eloszlású véletlen szám generálásához

Rövidítések		
ABC	artificial bee colony algorithm	mesterséges méhkolónia algoritmus
ALU	aritmetical and logical unit	aritmetikai és logikai egység
API	application programming interface	applikációs programozási felület
BBO	biogeography-based optimization	biogeográfia alapú optimalizáció
DE	differencial evolution algorithm	differenciális evolúció algoritmus
CHS	circular hollow section	körcső keresztmetszet
$CUDA$	compute unified device architecture	egységes számítási eszközarchitektúra
EA	evolutionary algorithm	evolúciós algoritmus
EC	evolutionary computation	evolúciós számítás
FA	firefly algorithm	szentjánosbogár algoritmus
FPA	flower pollination algorithm	virágbeporzási algoritmus
GA	genetic algorithm	genetikus algoritmus
$GMAW-C$	gas metal arc welding with CO_2	CO_2 védőgázas ívhegesztés
GPU	graphical processing unit	grafikus számítóegység (köznyelvben a teljes videó/grafikus kártya)
HS	harmony search algorithm	harmónia keresési algoritmus
IWO	invasive weed optimization	invazív gyom optimalizáció
$null$		üres mutató
$NVIDIA$		grafikus processzorokat gyártó vállalat
PSO	paricle swarm optimization	részecskeraj optimalizáció
RHS	rectangular hollow section	négyzetes cső keresztmetszet
$RMSE$	root-mean-square error	átlagos négyzetes hiba
$SaDE$	self-adaptive differential evolution algorithm	önadaptív differenciális evolúció algoritmus
$SaNSDE$	self-adaptive differential evolution with neighborhood search	önadaptív differenciális evolúció szomszédsági kereséssel
$SIMD$	single instruction multiple data	egy utasítás több adat
$SIMT$	single instruction multiple thread	egy utasítás több szál
SM	streaming multiprocessor	
$StdBA$	standard bee algorithm	szabványos méh algoritmus

1. fejezet

Bevezetés

1.1. Rövid történeti áttekintés

Az optimálási problémák mindenütt jelen vannak mindennapjainkban. Mikor a munkába menet kiválasztjuk az odavezető utat, mikor megpróbálunk bepakolni egy hátizsákba, vagy mikor kiválasztjuk a befektetéseinket a várható hozam maximalizálásának érdekében, akkor lényegében egy optimalizálási problémát oldunk meg. A természeti folyamatokban, állatoknál, növényeknél egy-egy probléma megoldásához nincs szükség formális képzésre. Az optimálási problémák gyors megoldása minden faj számára kulcsfontosságú a túléléshez, ezért az idők során ezt az evolúció előnyben részesítette. Nyilvánvaló, hogy ezek heurisztikusan kerülnek megoldásra, nem pedig pontosan, vagyis nem garantált, hogy az így előállított közelítő megoldások pontosak. Egy-egy újonnan felmerülő feladat lehetséges megoldását a természet megpróbálja egy múltbéli megoldásból levezetni. Ezt nevezik analógiával tanulásnak, és addig alkalmazódik iteratívan, míg el nem ér egy célállapotot, vagy a célállapothoz közelebbit.

A heurisztikák (és metaheurisztikák) a földi élet kialakulása óta jelen vannak, de a tudományos vizsgálatukhoz a XX. századig várni kellett [105]. Feltételezhetően annyira természetesek, hogy meg kellett várni az optimalizálás formális kialakulását. Történelmük öt különböző időszakra bontható [105], melyek között éles határ nem húzható a paradigmaváltások alkalmával:

- Tudományos kutatások előtti időszak körülbelül az 1940-es évekig, a heurisztikákat és metaheurisztikákat alkalmazták, de nem kerültek tanulmányozásra.
- Korai kutatási időszak 1940 és 1980 között, ekkor jelentek meg az első cikkek, formális tanulmányok.
- Módszer-, eljáráscentrikus időszak 1980 és 2000 között, a metaheurisztika területe fellendülőben van, ekkor publikálják a legtöbb új eljárást, sokféle új módszert javasolnak.
- A 2000-es évektől még napjainkban is tart a keretrendszerközpontú időszak, érdemesebb a metaheurisztikákat egy keretként leírni, mintsem önállóan egy-egy módszerként vagy eljárásként, figyelembe véve az alkalmazást is.

A napjainkig több mint 192 eljárást dokumentáltak. Az 1.1. ábra alapján a harmadik korszak, ellenben a [105] irodalmi adatokkal, inkább a 2000-es évek utáni időszak, és még

napjainkban is tart. Új módszerek és eljárások publikálása mellett megjelentek és egyre gyakoribbak azok az irodalmak, melyek az algoritmusok paramétereinek beállításával, vagy az eljárások pontosságának növelésével foglalkoznak. A paraméterek hosszadalmas kézi beállítását lassan felváltják az adaptív módszerek [3, 23, 96, 124]. Pontosság és hatékonyság növelését pedig fokozhatják a különböző eljárások egymással történő többszintű vagy párhuzamos kombinálása [63, 64].

A módszerek kutatásával szinte egyidőben megjelentek azok alkalmazásai és az azokra irányuló kutatások, publikációk. Számos területen alkalmazhatóak hatékonyan. A teljesség igénye nélkül néhányat kiemelve közülük: szerkezet optimalizálás [14, 55], legrövidebb út probléma [26], óceáni hullámerőművek optimális elhelyezése farmokon [79].

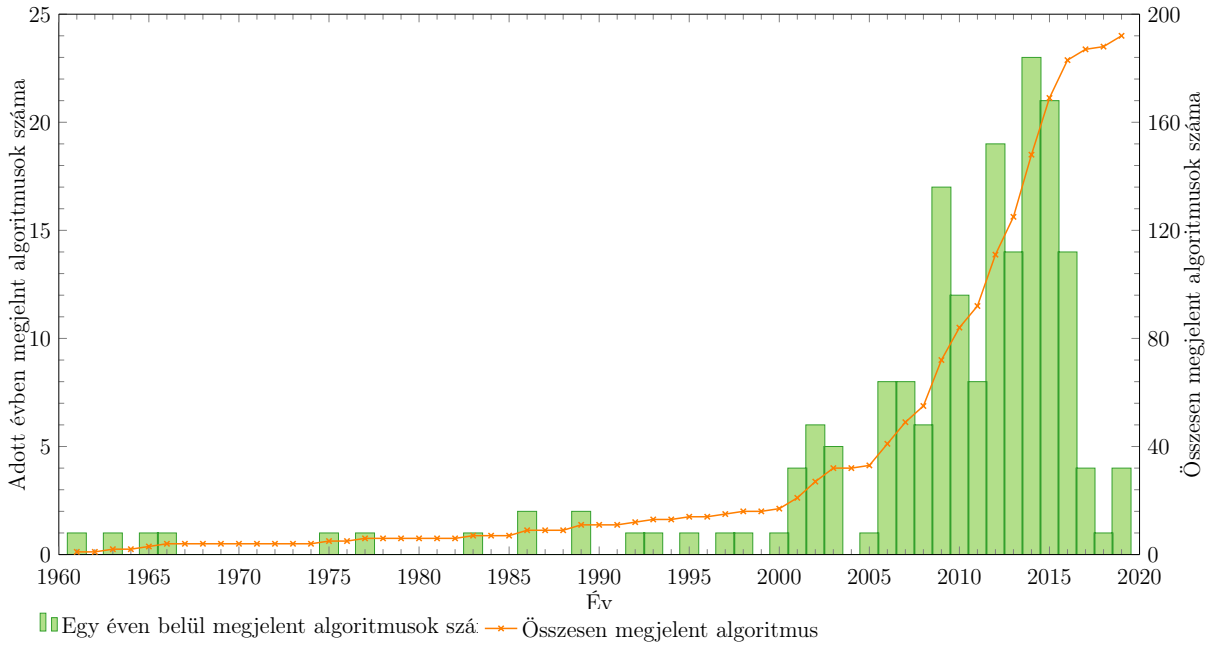
1.2. Irodalmi áttekintés

Az evolúciós számítások (EC) a mesterséges intelligencia részterületét képezik, azon belül is a sztochasztikus kereső eljárások részhalmaza. Ezek olyan iteratív módszerek és eljárások gyűjteménye, melyek a már korábbi eredményeket felhasználva folyamatosan fejlesztik a lehetséges megoldást, megoldások halmazát [25].

A működésüket elsősorban az újabb és újabb megoldásokat generáló matematikai modell határozza meg. Kialakításuk koncepciója többféle lehet, és többféleképpen lehet őket csoportosítani. Burgolya [21] szerint lehetséges csoportosítás:

– *biológiai evolúció ösztönözte módszerek*: evolúciós folyamatokban megfigyelt lépéseket (pl.: szelekció, öröklődés stb.) modellezik matematikai formalizmussal. Ide tartoznak a teljesség igénye nélkül a genetikus algoritmusok (GA) [5, 36], genetikus programozás, evolúciós algoritmusok (EA) [25], evolúciós programozás [34] és evolúciós stratégiák [6];

– *biológiai eljárás, viselkedés alapú módszerek*: ilyenek lehetnek például a különböző



1.1. ábra. Evolúciós algoritmusok számának alakulása az évek során [4]

raj-intelligenciák [2, 28, 29], fajok élelem keresési módszerei, szaporodási stratégiák és sok más egyéb biológiai viselkedés [16, 39, 75, 119, 126];

– *tisztán matematikai modelleket alkalmazó módszerek*: ebben az esetben elsősorban valamilyen absztrakt lineáris kombináció, vagy valószínűségi modell alkalmazására kell gondolni [38, 96, 107].

Fontos megjegyezni, hogy a szakirodalomban sokszor az evolúciós számításokat (EC) és evolúciós algoritmusokat (EA) egymás szinonimájaként használják. Holott az EA az EC egy részhalmaza. A továbbiakban ezen szinonimaként értelmezést alkalmazom.

Szabó [108, 109] irodalmakban módszert javasol a szükséges iteráció szám (kilépési feltétel) becslésére a konvergencia és iterációs történet elemzésével. Lehetővé téve a sztochasztikus működésből adódó, feladatonként eltérő konvergencia becslését.

A napjainkban folyamatosan jelentkező alapanyag és energia árak emelkedése, a környezetvédelmi irányelvek szigorodása indokolja és szükségessé teszi az optimális gépészeti megoldások, szerkezetek kialakítását. Mankovits és társai gumi rugó alak-optimálását végez el végeselem-módszert alkalmazva a [43, 68, 69, 70] irodalmakban. Érintkezési feladatokhoz köthető optimálási problémát old meg Baksa és Páczelt a [9, 10] munkájukban, melyhez végeselem-módszert alkalmaznak. Ezeknél a feladatoknál az optimálásnak két lehetséges kimenete lehet. Egyrészt a kinematikai mennyiségek [82], míg másrészről a dinamikai mennyiségek [85, 86] optimálása. Érintkezési feladatok végeselemes formalizmusával foglakozik Baksa és társai [8, 12, 88].

Szerkezet optimáláshoz kapcsolódóan rácsos tartójú távvezeték tornyok optimális kialakítását vizsgálta Rao [95, 97] és Silva és társai [24]. Ezeknél a szerkezeteknél általában szögacél-szelvényeket alkalmaznak Taniwaki [110], de kis kihajlási merevségük miatt érdemesebb körcső-szelvényeket alkalmazni Orbán és társai [80] szerint. A Távvezeték tornyok tönkremenetelével foglakozik Rao [98].

Virág és Jármai [113] irodalomban különböző bordakialakítások mellett tekinti át bordázott lemezek optimális kialakítását. Virág és Szirbik [114, 115, 116] munkáikban végeselem-modell segítségével vizsgálják optimált bordáslemez paraméterváltozásainak hatását a sajátértékekre. Egytengelyű terhelést feltételezve, a numerikus vizsgálat során bordákat eltávolítva a stabilitásvesztéshez tartozó sajátértékeket határozzák meg.

Hegesztett szerkezetek tervezése során az egyik legkritikusabb probléma a stabilitási kérések. Minimális tömegű szekrényszelvényű oszlopok tervezésére mutat be módszert Petrik és társai [90], megvizsgálják több szabványi előírás és eltérő folyáshatárú acélok alkalmazásának hatását a tömegminimumra. A tűzhatás hasonlóan kritikus tervezési szempont lehet, mint a stabilitásvesztés. Petrik és társai módszert ismertet nyomástartó edények optimális kialakítására tűzterhelés esetén a [91] irodalomban.

Szerkezetköltség szempontjából optimális kialakítása során a költség függvények meghatározása nehéz, mert időben változnak Klansek és Kravanja [62], Jalkanen [46], és függenek az adott ország adottságaitól Tímár és társai [44]. Összehasonlítás céljából alapul kell venni a nemzetközileg mért gyártási időket, adatokat, és ezeket kell felszorozni egy szélesebb tartományban változtatható költségtényezővel Jármai és társai [49, 53, 54]. Hegesztési időkkel és költségekkel foglakozik Pahl és Beelich [57], továbbá Hubka [81]. Egyéb költségeket – úgymint festési, darabolási, lemezegyengetési költség stb. – ismertet Jármai [49] könyvében és Farkas és Jármai [47, 48] művekben.

1.3. Célkitűzések

1. **Cél**: A dolgozatban vizsgált evolúciós algoritmusokhoz, úgymint mesterséges méh kolónia (ABC) [56, 58, 59], méh algoritmus (StdBA) [93, 94], biogeográfia alapú optimalizáció (BBO) [13, 102, 103], differenciális evolúció (DE) [107], adaptív differenciális evolúció (SaDE és SaNSDE) [96, 124], szentjánosbogár algoritmus (FA) [119, 121], virágbeporzási algoritmus (FPA) [120, 123], harmónia keresés (HS) [35, 99], invazív gyom optimalizáció (IWO) [66, 76], és a részecskeraj optimalizáció (PSO) [18, 101] köthető szakirodalmak egymással nehezen vagy egyáltalán nem összehasonlítható teljesítmény eredményeket közölnek. A teljesítmény értékeléshez használt tesztfüggvények, vagy a szimulációs környezeti beállítások eltérnek.

   Célom a felsorolt algoritmusok széles körű szimulációja folytonos tesztfüggvényekkel és egységes környezeti beállításokat használva. Azok rangsorolása a szimulációs eredmények alapján.

2. **Cél**: Napjainkban a grafikus kártyák számítási kapacitása felhasználható általános célú számításokhoz is [22, 42, 51]. Ezzel lehetőséget biztosítva a nagy számítás igényű párhuzamos SIMD és SIMT architektúrát követő adat feldolgozásnak. Elérhetőek kifejezetten a CUDA API-t használó elemi, lineáris algebrához köthető és numerikus algoritmusok párhuzamos változatai [61]. Néhány esetben ezek futásidő szerinti optimalizált változatuk is, mint például a párhuzamos redukció [72].

   Célom a grafikus kártya adta lehetőségek kihasználásával módszert találni az FPA algoritmus és az általa optimált fitnesz függvény párhuzamos végrehajtására. Különös figyelmet fordítok a párhuzamos redukcióval kiszámítható célfüggvényekre. Célom még a normál szekvenciális futás és a párhuzamos futtatás közötti számítási kapacitások feltárása.

3. **Cél**: Az optimáláshoz köthető kutatások [1, 65, 70, 90, 114] egyre jelentősebbek. Ezen belül is jelentős szerepet kapnak az evolúciós algoritmusok alkalmazásai a szerkezetoptimálásban [14, 54, 55, 49].

   Célom két gyakorlatban előforduló szerkezet optimálása evolúciós módszerekkel, úgy, mint a kisteherautó-„plató” kereszttartóinak és a futódaru főtartójának optimálása. A teherautó plató esetén megvizsgálom, hogyan változik az optimált tömege, ha eltérünk az eredeti szerkezetben alkalmazott keresztmetszet geometriától, és változik a kereszttartók száma. Futódaru-főtartójának optimálása során megvizsgálom a költségek változását, más és más horogteher, fesztáv és alapanyagok használata mellett.

4. **Cél**: A végeselem-módszer egy különböző variációs elvekre támaszkodó közelítő számítási módszer [15, 30, 43, 83, 87]. Húzott-nyomott rudakkal modellezett rácsos tartók esetén ez a közelítő számítás az egzakt megoldást adja [83].

   Célom módszert találni az evolúciós optimálás és a húzott-nyomott rúdelemekkel modellezhető szerkezetek végeselemes megoldásának összekapcsolására. A talált módszerrel gyakorlatban előforduló feladat megoldása. A választott számszerűsített mérnöki probléma egy rúdelemekkel modellezhető, rácsos tartó szerkezetű, csonka gúla alakú távvezeték torony alsó részének optimálása. Az optimálás során megvizsgálásra kerül, hogy miként változik a szerkezet tömege, ha az alapanyag folyáshatára $f_y = 235\,\mathrm{MPa}$ és $f_y = 690\,\mathrm{MPa}$ határok között, vagy a rácsosztások száma változik,

vagy ha korlátozzuk az eredetileg legnagyobb elmozdulással rendelkező csomópont elmozdulását.

2. fejezet

Alkalmazott eljárások, módszerek és modellek

2.1. Evolúciós algoritmusok

Az algoritmusok bemutatásához és tárgyalásához elengedhetetlen az alábbi fogalmak definiálása [21, 25, 36]:

- *populáció*: A feladat összes lehetséges megoldását tartalmazó $\mathcal{S}^D$ keresési tér $\mathcal{P}$ valódi részhalmaza. A $\mathcal{P}$ halmaz számosságával definiálható az n_p populációméret,

$$\mathcal{P} \subset \mathcal{S}^D \qquad |\mathcal{P}| = n_p \qquad 0 < n_p < \infty\,. \tag{2.1}$$

- *egyed*: A $\mathcal{S}^D$ keresési tér egy eleme. Kitüntetett szerepük van azoknak ez egyedeknek, melyek egyben a populáció részét is képezik és külön tárolásra kerülnek.
- *fitnesz-függvény*: A $\mathcal{S}^D$ keresési térben értelmezett függvény, mely minden egyes megoldás jóságát tükrözi. Alkalmas az egyedek rangsorolására.
- *generáció*: Időben együtt létező egyedek a populációban.
- *generációs ciklus*: Egy iterációs lépésben alkalmazott evolúciós műveletek.

Az 1. algoritmus foglalja össze általánosan egy EA felépítését. A feltüntetett lépések konkrét implementációját mindig az adott algoritmus publikációja tartalmazza. Egy eljáráson belül nem feltétlen szükséges az összes műveletet megvalósítani, de az is előfordulhat hogy egy-egy művelet többször is előfordul egy iterációs lépésen belül (például: [59]).

Algoritmus 1: EA általános felépítése

```
1 (0)P populáció inicializálása
2 while kilépési feltétel hamis do
3     szelekció
4     rekombináció
5     mutáció
6     visszahelyezés
```

Szelekció művelet során cél a szülőpopuláció kiválasztása, törekedve az átlagos fitnesz érték javítására, de legalábbis annak szinten tartására

$$^{(G)}\mathcal{P}_p \subseteq {}^{(G-1)}\mathcal{P}\,, \tag{2.2}$$

ahol $\mathcal{P}_p$ szülő egyedek halmaza. Működésének az alapja, hogy a szülő egyedek, és a gyerek-egyedek fitnesz értékei között korreláció mutatható ki [21]. Az egyed jóságának függvényében egy-egy egyed akár többször is belekerülhet a $^{(G)}\mathcal{P}_p$ populációba, vagy egyszer sem. *Rekombináció* vagy *keresztezés* művelet során egy vagy több szülő képez utódot. Mivel az utód a szülők környezetében generálódik, az új pont a szülők által kijelölt / határolt keresési téren belül lesz [17]. Az iteratív lépések során ez a tér folyamatosan szűkül. Lényegében ez az a művelet, ami élesen megkülönbözteti a különböző evolúciós algoritmusokat, és számos formája létezik. Rekombináció művelete lehetővé teszi a lokális, illetve a globális keresést, de nem igazán alkalmas a finomabb közelítésekre, vagy hogy egy-egy utód „kitekintsen" a szülők által kijelölt hiperkockából [21]. Erre való a *mutáció*, ami az utód szűk közvetlen környezetében keres

$$\left\{ {}^{(G+1)}\mathbf{x}_i \in \mathcal{S}^D \mid d\left({}^{(G+1)}\mathbf{x}_i, {}^{(G)}\mathbf{x}_i \right) < \varepsilon \right\}\,, \tag{2.3}$$

ahol $^{(G)}\mathbf{x}_i$ a mutáció előtti $^{(G+1)}\mathbf{x}_i$ pedig a mutáció utáni egyed, $d(\ ,\ )$ az Euklideszi távolság két egyed között, ε a közvetlen környezet mértéke. Az ε értéke vagy konstans, vagy folyamatosan csökken az iteráció előrehaladtával (például: [121]). A konkrét megvalósítás algoritmusonként eltérő, de a leggyakoribb implementáció az ε sugarú körön belüli véletlen mozgás. Végül a *visszahelyezés* során biztosítani kell, hogy a szelekcióval, rekombinációval, mutációval létrejött és a következő iteratív lépésbe átvitt populáció mérete ne haladja meg a rögzített n_p értéket. Fontos jellemzője, hogy a meglévő populáció egyedeinek hányad részét cseréljük le újakkal (visszahelyezési ráta).

Az EA ciklus kialakításánál a kilépési feltétel megfogalmazása feladatfüggő, gyakran mindig valami idővel arányos megfogalmazás. A teljesség igénye nélkül lehet: maximális generációszám, maximális futási idő, maximális függvényérték kiszámítás, stb. Egy adott érték megközelítésétől, mint kilépési feltételtől óvakodni kell. Túl szigorú esetben, és ha a fitnesz függvény túl zajos, elképzelhető, hogy sose közelíti meg, mert „beragad" egy lokális optimumba.

2.1.1. Biogeográfia alapú optimálás

Biogeográfia alapú optimálást (BBO) először [102] javasolta. Biogeográfiai folyamatok és fogalmak inspirálták, úgymint: az új fajok kialakulása, a fajok kihalása és a fajok szigetek közötti vándorlása.

Az életbarát szigetek élőhely alkalmassági indexszel (HSI) rendelkeznek. Az életbarát tulajdonságokat befolyásoló tényezőket alkalmassági indexnek szokás nevezni (SVI). Az optimalizálás során lakhatóság szempontjából az SVI-k a független változók, míg a HSI a függő változónak feleltethető meg. A magas HSI-vel rendelkező szigeten sok faj található meg, amiről a közeli szigetekre történő μ_k elvándorlásnak nagy a valószínűsége a túlzsúfoltság miatt. Ez azt is jelenti, hogy alacsony a bevándorlási λ_k valószínűség ezekre a szigetekre, hiszen túlzsúfoltak. Az alacsony HSI-vel rendelkező szigeteken ezen folyamat ellentettje történik. A leírt emigrációs, immigrációs folyamatokat modellezi a 2. algoritmus, és használja a BBO. A gyakorlatban a μ_k elvándorlási valószínűség megegyezik a rulettkerék szelekcióhoz köthető valószínűséggel.

Algoritmus 2: Biogeográfia alapú optimalizáció pszeudokódja

szükséges konstansok beállítása: p_p, p_{pa}, α_{damp}
${}^{(0)}\mathcal{P}$ populáció ${}^{(0)}\mathbf{x}_i$ egyedeinek inicializálása
while *kilépési feltétel hamis* **do**
  for $i \in [1, n_p]$ **do**
    $\mu_{k,i}$ kivándorlási valószínűség rendelése ${}^{(G)}\mathbf{x}_i$ egyedhez, $\mu_{k,i} \propto \mathcal{F}({}^{(G)}\mathbf{x}_i)$
    $\lambda_{k,i}$ bevándorlási valószínűség rendelése ${}^{(G)}\mathbf{x}_i$ egyedhez, $\lambda_{k,i} = 1 - \mu_{k,i}$
  for $i \in [1, n_p]$ **do**
    for $j \in [1, n_D]$ **do**
      k indexű egyed választása ${}^{(G)}\mathcal{P}$ populációból rulettkerék szelekcióval
      if $\mathcal{U}(0,1) < \lambda_{k,i}$ **then**
        ${}^{(G+1)}x_{i,j} = {}^{(G)}x_{i,j} + F\left({}^{(G)}x_{k,j} - {}^{(G)}x_{i,j}\right)$
    if $\mathcal{U}(0,1) \leq p_p$ **then**
      ${}^{(G+1)}\mathbf{x}_i$ véletlenszerűen mozog
    if $\mathcal{F}({}^{(G+1)}\mathbf{x}_i) < \mathcal{F}({}^{(G)}\mathbf{x}_i)$ **then**
      ${}^{(G+1)}\mathbf{x}_i$ bekerül ${}^{(G+1)}\mathcal{P}$ populációba
    else
      ${}^{(G)}\mathbf{x}_i$ bekerül ${}^{(G+1)}\mathcal{P}$ populációba
  utófeldolgozás, eredmények vizualizációja

2.1.2. Differenciális evolúció és változatai

Differenciális evolúció (DE) [107] az egyik legrégebbi eljárás, mely bináris műveletek helyett valós vektorokon végez műveleteket. Egyes irodalmak [17, 25] szerint ez tekinthető az első tényleges evolúciós algoritmusnak. A optimálási feladatok megoldása során sokkal hatékonyabbnak bizonyult ez a fajta megközelítés, mint a csak bináris műveleteket alkalmazó eljárások [107].

A $\mathcal{S}$ keresési térben a lehetséges megoldásokat iteratív módon n_p egyedet számláló populáció $\mathbf{x}_i$ egyedein

$$\mathbf{x}_i \in \mathcal{S} \qquad i \in [1, n_p] \tag{2.4}$$

végzett mutáció, keresztezés és visszahelyezés műveletek segítségével keresi.

Mutáció során a G generációs ${}^{(G)}\mathbf{x}_i$ egyedeket felhasználva képez egy mutáns ${}^{(G)}\mathbf{v}_i$ vektort. A teljesség igénye nélkül, a gyakorlatban legnépszerűbben alkalmazott változatok [107, 96, 124]:

– DE/rand/1:

$${}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_i + F_S\left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2}\right), \tag{2.5}$$

– DE/best/1:

$${}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_{best} + F_S\left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2}\right), \tag{2.6}$$

– DE/current to best/1:

$${}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_i + F_S\left({}^{(G)}\mathbf{x}_{best} - {}^{(G)}\mathbf{x}_i\right) + F_S\left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2}\right), \tag{2.7}$$

– DE/best/2:

$$ {}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_{best} + F_S \left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2} \right) + F_S \left({}^{(G)}\mathbf{x}_{r_3} - {}^{(G)}\mathbf{x}_{r_4} \right) , \tag{2.8} $$

– DE/rand/2:

$$ {}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_{r_1} + F_S \left({}^{(G)}\mathbf{x}_{r_2} - {}^{(G)}\mathbf{x}_{r_3} \right) + F_S \left({}^{(G)}\mathbf{x}_{r_4} - {}^{(G)}\mathbf{x}_{r_5} \right) , \tag{2.9} $$

ahol $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \in [1, n_p]$ véletlen indexek, $F_S \in (0,2]$ skálázási tényező, és $\mathbf{x}_{best}$ a G. iterációs lépésben a legjobb fitnesz értékkel rendelkező egyed.

A mutációs művelet során létrehozott ${}^{(G)}\mathbf{v}_i$ mutáns vektor és a ${}^{(G)}\mathbf{x}_i$ egyed páronként kerülnek keresztezésre, az így létrehozott ${}^{(G)}\tilde{\mathbf{u}}_i$ vektorok a próbavektorok. A keresztezés művelet célja a j vektorkoordináták diverzitásának növelése

$$ {}^{(G)}\tilde{\mathbf{u}}_{i,j} = \begin{cases} {}^{(G)}\mathbf{v}_{i,j} & \text{ha } \mathcal{U}(0,1) \leq C_R \text{ vagy } j = j_{rand} \\ {}^{(G)}\mathbf{x}_{i,j} & \text{egyébként} \end{cases} , \tag{2.10} $$

ahol $C_R \in [0,1)$ keresztezési arány, $j_{rand} \in [1, n_D]$ véletlen vektorkoordináta index.

Visszahelyezés (egyes irodalmak szerint szelekció) során a következő generációba csak azok a ${}^{(G)}\tilde{\mathbf{u}}_i$ próba vektorokkal jellemzett egyedek kerülnek be, melyek fitnesz értéke jobb a már a populáció részét alkotó ${}^{(G)}\mathbf{x}_i$ egyedekénél

$$ {}^{(G+1)}\mathbf{x}_i = \begin{cases} {}^{(G)}\tilde{\mathbf{u}}_i & \text{ha } \mathcal{F}\left({}^{(G)}\tilde{\mathbf{u}}_i \right) < \mathcal{F}\left({}^{(G)}\mathbf{x}_i \right) \\ {}^{(G)}\mathbf{x}_i & \text{egyébként} \end{cases} . \tag{2.11} $$

A DE optimálási teljesítményét nagy mértékben befolyásolja az alkalmazott mutációs stratégia és a F_S, C_R paraméterek megválasztása. Ezek feladatonként eltérnek. A megválasztásuk sokszor fáradtságos és időigényes feladat. A [96] irodalom önadaptív módszert ajánl, mely lehetővé teszi felhasználói beavatkozás nélkül az algoritmus beállítását. Így bevezetett evolúciós módszer neve: önadaptív differenciális evolúció (SaDE).

A mutációs lépés során (2.5) és (2.7) egyenletekkel definiált stratégiák között választ egy p_p valószínűségi változó segítségével

$$ {}^{(G)}\mathbf{v}_i = \begin{cases} {}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_i + F_S \left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2} \right) & \text{ha } p_p \leq \mathcal{U}(0,1) \\ {}^{(G)}\mathbf{v}_i = {}^{(G)}\mathbf{x}_i + F_S \left({}^{(G)}\mathbf{x}_{best} - {}^{(G)}\mathbf{x}_i \right) + F_S \left({}^{(G)}\mathbf{x}_{r_1} - {}^{(G)}\mathbf{x}_{r_2} \right) & \text{egyébként} \end{cases} , \tag{2.12} $$

ahol F_S skálázási tényező minden egyes mutációs műveletnél egyedenként új értéket kap, és $F_S = \mathcal{N}(0{,}5; 0{,}3) \in [0,2)$. A valószínűségi változó

$$ p_p = \frac{n_{s1} \left(n_{s2} + n_{f2} \right)}{n_{s2} \left(n_{s1} + n_{f1} \right) + n_{s1} \left(n_{s2} + n_{f2} \right)} \tag{2.13} $$

összefüggés szerint n_{lp} iterációs lépés (tanulási ciklus) után frissül, ahol n_{s1}, n_{s2} (2.5) illetve (2.7) egyenletekkel generált és a $(G+1)$. generációba sikeresen visszahelyezett egyedek száma, és n_{f1}, n_{f2} (2.5) illetve (2.7) szerint generált, de a populációba vissza nem helyezett egyedek száma. Minden egyes tanulás lezárásaként n_{s1}, n_{s2}, n_{f1} és n_{f2} számlálókat nullázni szükséges. Az első tanulási ciklus során a valószínűségi változó javasolt kezdő értéke $p_p = 0{,}5$. Előfordulhat, hogy (2.13) nevezője zérus, ha nem volt a n_{lp} számú ciklus során egyetlen egy sikeres visszahelyezés sem. A [96] irodalom erre az eshetőségre nem tér ki. Az algoritmus felhasználása során erre az esetre saját módszert vezettem be, melynek lényege,

hogyha (2.13) nevezője zérus, akkor p_p valószínűségi változó nem frissül a n_{lp} ciklus után, és a számlálók értéke se nullázódik mindaddig, míg legalább egy sikeres visszahelyezés nem történik.

A C_R keresztezési arány n_{CR} iterációs lépésenként frissül véletlenszerűen

$$C_R = \mathcal{N}(C_{Rm}; 0{,}1) \in [0{,}1)\,, \tag{2.14}$$

ahol C_{Rm} azon C_R értékek átlaga, melyekkel generált egyedek sikeresen bekerültek a következő $(G+1)$ generációba egy n_{lCR} iterációs lépésből álló tanulási ciklus során. Itt is előfordulhat, hogy a C_{Rm} érték zérus, ha nincs sikeres visszahelyezés, amit a [96] nem kezel le. Az alkalmazott saját módszer, hasonló az előzőhöz, jelen esetben a C_{Rm} érték nem frissül mindaddig, míg nincs sikeres visszahelyezés.

A szomszédsági keresés [125] és az ön-adaptivitás előnyeit egyesíti a [124] irodalom által javasolt SaNSDE algoritmus. A [96] irodalomban bemutatott C_{Rm} meghatározásához használt egyszerű átlagolás helyett bevezet egy új, súlyozott átlagolást

$$C_{Rm} = \sum_{k=1}^{|\mathbf{C}_{Rrec}|} \omega_k C_{Rrec,k}\,, \tag{2.15}$$

$$\omega_k = \frac{\Delta\mathcal{F}_{rec,k}}{\sum_{k=1}^{|\Delta\mathcal{F}_{rec}|} \Delta\mathcal{F}_{rec,k}}\,, \tag{2.16}$$

ahol $\mathbf{C}_{Rrec}$ a n_{lCR} hosszú tanulási ciklus alatt a sikeresen visszahelyezett egyedhez tartozó C_R keresztezési arányokból képezett vektor

$$\mathbf{C}_{Rrec} = \begin{bmatrix} C_{Rrec,1} & C_{Rrec,2} & C_{Rrec,3} & \dots & C_{Rrec,k} & \dots & C_{Rrec,n} \end{bmatrix}^T\,, \tag{2.17}$$

$\Delta\mathcal{F}_{rec}$ vektor pedig a sikeres visszahelyezésekhez tartozó fitnesz érték változásokból képezett vektor

$$\Delta\mathcal{F} = \begin{bmatrix} \Delta\mathcal{F}_{rec,1} & \Delta\mathcal{F}_{rec,2} & \Delta\mathcal{F}_{rec,3} & \dots & \Delta\mathcal{F}_{rec,k} & \dots & \Delta\mathcal{F}_{rec,n} \end{bmatrix}^T\,. \tag{2.18}$$

A két vektor elemszáma azonos $n = |\mathbf{C}_{Rrec}| = |\Delta\mathcal{F}_{rec}|$. A tanulási ciklus végén mind a $\mathbf{C}_{Rrec}$, mind a $\Delta\mathcal{F}_{rec}$ vektorokat törölni kell. A [124] irodalom sem ad javaslatot arra az esetre, ha a C_{Rm} értéke zérus. Az általam alkalmazott módszer megegyezik a SaDE-nél leírtakkal.

2.1.3. Harmónia keresési algoritmus

Harmónia keresési algoritmust (HS) először [35] irodalom publikálta. A jazz zenészek viselkedése inspirálta. Zenei harmóniát létrehozva saját játékukat fokozatosan a zenekarhoz igazítják. Hamis hang esetén improvizációval javítják előadásukat.

Az algoritmussal foglakozó irodalmak [35, 99] sajátos analógiát vezetnek be az eddig ismertetett terminusokra. A $\mathcal{P}$ populációt harmónia memóriának, az ${}^{(G)}\mathbf{x}_i$ egyedeket harmóniának, míg az egyedek ${}^{(G)}x_{i,j}$ döntési változóit hangmagasságnak, hangjegynek és végül n_{IT} maximális iteráció számot improvizációnak nevezik.

Új harmóniák (egyedek) létrehozása történhet a harmónia memóriából közvetlenül, vagy a harmónia memóriából kisebb módosítással, vagy végül teljesen véletlenszerűen a keresési térben. A HS működését a 3. algoritmus írja le, ahol p_p egy valószínűségi változó (konszolidációs ráta), p_{pa} hangjegy hangolási ráta, $\alpha_{damp} \in [0{,}1]$ skálázási konstans, és ${}^{(G)}\mathbf{x}_{worst}$ a G. populációban a legrosszabb fitnesz értékű egyed.

Algoritmus 3: Harmónia keresési algoritmus pszeudokódja

szükséges konstansok beállítása: p_p, p_{pa}, α_{damp}
${}^{(0)}\mathcal{P}$ populáció ${}^{(0)}\mathbf{x}_i$ egyedeinek inicializálása
while *kilépési feltétel hamis (ált. maximális improvizáció szám)* **do**
  for $i \in [1, n_p]$ **do**
    for $j \in [1, n_D]$ **do**
      if $\mathcal{U}(0,1) \leq p_p$ **then**
        ${}^{(G+1)}x_{i,j}$ legyen ${}^{(G)}\mathcal{P}$ véletlenszerűen választott egyedének j. hangjegye
        if $\mathcal{U}(0,1) \leq p_{pa}$ **then**
          ${}^{(G+1)}x_{i,j} = {}^{(G+1)}x_{i,j} + \alpha_{damp}\,\mathcal{U}(0,1)(x_{U,j} - x_{L,j})$
      else
        ${}^{(G+1)}x_{i,j} = \mathcal{U}(x_{L,j}, x_{U,j})$
    if $\mathcal{F}({}^{(G+1)}\mathbf{x}_i) < \mathcal{F}({}^{(G)}\mathbf{x}_{worst})$ **then**
      ${}^{(G+1)}\mathbf{x}_i$ bekerül a ${}^{(G+1)}\mathcal{P}$ populációba
    else
      ${}^{(G+1)}\mathbf{x}_i$ eldobásra kerül, és ${}^{(G+1)}\mathcal{P}$ populációba ${}^{(G)}\mathbf{x}_i$ kerül be
utófeldolgozás, eredmények vizualizációja

2.1.4. Invazív gyom optimalizáció

Az invazív gyom optimalizáció (IWO) természet ihlette eljárás, melyet [73] irodalomban publikáltak először. A gyomok szaporodási, terjedési stratégiáját alkalmazza. A gyomok olyan növénykultúrák, melyek populációja földrajzi helyzettől függetlenül zavarta, zavarja az embert. A gyomok veszélyes és agresszív növények a mezőgazdaságban, hiszen több ezeréves talajművelés, 50 évnyi gyomirtózás következtében is évről évre visszatérnek a termőföldekre. Egyetlen egy gyomfaj se tűnt el vagy halt ki. Ezen tulajdonságok azt erősítik meg, hogy időről időre ezek a növények egyre ellenállóbbá, fittebbé válnak.

Az iterációs lépések során a gyomok (egyedek) véletlenszerűen szétszórják magjaikat a szülő egyed környezetében

$$ {}^{(G+1)}\mathbf{x}_i = \left[\mathcal{N}({}^{(G)}x_{i,1}, {}^{(G)}\,\sigma_1), \mathcal{N}({}^{(G)}x_{i,j}, {}^{(G)}\,\sigma_j), \ldots, \mathcal{N}({}^{(G)}x_{i,n_D}, {}^{(G)}\,\sigma_{n_D})\right]^T , \tag{2.19} $$

ahol ${}^{(G)}\sigma_j$ a szórás iterációs lépésről iterációs lépésre változik

$$ {}^{(G)}\sigma_j = \frac{(n_{IT} - G)^n}{(n_{IT})^n} \left(\sigma_{init,j} - \sigma_{final,j}\right) + \sigma_{final,j} , \tag{2.20} $$

ahol n nemlinearitást biztosító kitevő, $\sigma_{init,j}$ és $\sigma_{final,j}$ egy kezdő és vég értéke σ_j szórásnak. Minden egyed n_{SR} számú egyedet hozz létre generációnként. A ${}^{(G+1)}(P)$ populációba az újonnan létrehozott ${}^{(G+1)}\mathbf{x}_i$ egyedek közül és a már meglévő ${}^{(G)}\mathcal{P}$ populáció egyedei közül a legfittebbek kerülnek be.

2.1.5. Méh és mesterséges méhkolónia algoritmus

A méh algoritmus (StdBA) [92] és a mesterséges méhkolónia algoritmus (ABC) [59] a méhek nektárgyűjtési szokását modellezi és alkalmazza folytonos függvények szélsőérték

kereséséhez. A modellben a méhek három csoportba sorolhatóak: alkalmazottak, bámészkodók és felderítők. Feltételezzük, hogy a méhek száma megegyezik a táplálékforrások számával. Az alkalmazottak a kaptárt elhagyva begyűjtik az ismert nektárlelőhely és annak környezetéből a nektárt, majd a kaptárba visszatérve, táncukkal jelzik a talált élelem mennyiségét és távolságát. A bámészkodó méhek a táncot figyelve döntenek, hogy mely élelemforrást aknázzák ki. A felderítők pedig véletlenszerűen mozognak a keresési térben, függetlenül a többi méh táncától.

Algoritmus 4: StdBA algoritmus pszeudokódja

```
szükséges konstansok beállítása: n_rb, n_rs, n_ob, n_os, r_1 és r_2
^(0)P populáció ^(0)x_i egyedeinek inicializálása
^(0)P populáció egyedeinek sorbarendezése fitnesz értékük alapján
while kilépési feltétel hamis do
    for i ∈ [1, n_rb] do
        for j ∈ [1, n_rs] do
            ^(G,i)P_w1 halmaz részeként dolgozó méhek létrehozása ^(G−1)x_i egyedből
             (2.23) egyenlet alapján
        ^(G)x_i egyed legyen a ^(G,i)P_w1 populáció legfittebb egyede, de csak abban az
         esetben ha ^(G,i)P_w1-ben létezik ^(G−1)x_i-nél fittebb egyed, egyébként
         ^(G)x_i = ^(G−1)x_i
    for i ∈ [n_rb + 1, n_rb + n_ob] do
        for j ∈ [1, n_os] do
            ^(G,i)P_w2 halmaz részeként dolgozó méhek létrehozása ^(G−1)x_i egyedből
             (2.23) egyenlet alapján
        ^(G)x_i egyed legyen a ^(G,i)P_w2 populáció legfittebb egyede, de csak abban az
         esetben ha ^(G,i)P_w2-ben létezik ^(G−1)x_i-nél fittebb egyed, egyébként
         ^(G)x_i = ^(G−1)x_i
    for i ∈ [n_rb + n_ob + 1, n_p] do
        ^(G)x_i legyen egy véletlen pont a keresési térben
    ^(G)P populáció egyedeinek sorba rendezése fitnesz értékük alapján
    ^(G)P populációból legjobb fitnesz értékű egyed kiválasztása
    r_1 és r_2 sugarak csökkentése
utófeldolgozás, eredmények vizualizációja
```

A StdBA és az ABC felépítése hasonló, melyet a 4. és az 5. algoritmusok szemléltetnek. Mind a két esetben a ${}^{(G)}\mathcal{P}$ populáció három részre van osztva: n_{rb} számú dolgozóra, n_{ob} számú bámészkodó méhre és végül n_{sb} számú felderítőre. A három csoport egyedszámára igaz

$$n_{rb} \leq n_p\,, \quad n_{ob} \leq n_p\,, \text{ és } \quad n_{sb} \leq n_p \tag{2.21}$$

továbbá ez a feltétel StdBA esetén kiegészül az alábbi feltétellel is

$$n_{rb} + n_{ob} + n_{sb} = n_p\,. \tag{2.22}$$

A méhek táncának matematikai modelljére a [92] és a [59] irodalmak javaslata megegyezik

$${}^{(G+1)}x_{i,j} = {}^{(G)}x_{i,j} + r\mathcal{U}(x_{L,j}, x_{U,j}) \qquad j \in [1, n_D]\,, \tag{2.23}$$

Algoritmus 5: ABC algoritmus pszeudokódja

```
1  szükséges konstansok beállítása: n_rb, n_ob, r, küszöbérték 1-1 egyed élethosszára
2  (0)P populáció (0)x_i egyedeinek inicializálása
3  while kilépési feltétel hamis do
4      for i ∈ [1, n_rb] do
5          (G)x_i egyed generálása (G−1)x_i egyedből (2.23) egyenlet alapján
6          if F((G)x_i) ≤ F((G−1)x_i) then
7              (G)x_i bekerül a (G)P populációba
8          else
9              (G−1)x_i bekerül a (G)P populációba
10             i. egyed élethosszának növelése
11     Rulettkerék kiválasztással (G)P populációból n_ob egyedszámmal (G)P_ob
         populáció képzése
12     for i ∈ (G)P_ob do
13         (G)x_i egyed generálása (G−1)x_i egyedből (2.23) egyenlet alapján
14         if F((G)x_i) ≤ F((G−1)x_i) then
15             (G)x_i bekerül a (G)P populációba
16         else
17             (G−1)x_i bekerül a (G)P populációba
18             i. egyed élethosszának növelése
19     for i ∈ (G)P do
20         if i. egyedélethossza > küszöb érték then
21             (G)x_i legyen egy véletlen pont a keresési térben
22     (G)P populációból legjobb fitnesz értékű egyed kiválasztása
23     r sugár csökkentése
24 utófeldolgozás, eredmények vizualizációja
```

ahol r egy konstans, mely kifejezi annak a körnek a sugarát, amin belül a dolgozó és bámészkodó méhek megvizsgálják a lehetséges nektárforrásokat a már ismert források környezetében.

A két algoritmus között a különbséget a különböző csoportokba sorolt méhek viselkedésének absztrakciója adja. StdBA esetén minden egyes dolgozó megvizsgál n_{rs} darabszámú lehetséges új nektárforrást az n_{rb} darabszámú legígéretesebb forrás környezetében, és visszatér a kaptárba a legjobbal (lásd 4. algoritmus 5–8 sora). Ezzel ellentétben az ABC dolgozói csak egyetlen egy új lehetségeset vizsgálnak meg (lásd 5. algoritmus 4–10 sora), és nincs megkötés a forrás minőségére.

A bámészkodó méhek StdBA esetén olyan lelőhelyeket látogatnak meg, melyek potenciálisan rosszabbak, mint a dolgozók által meglátogatottak. A dolgozókhoz hasonlóan n_{os} darabszámú esetet vizsgálnak meg és térnek vissza a legjobbal (lásd 4. algoritmus 9–12 sora). Gyakorlatban a dolgozók által feltérképezett r_1 sugarú kör kisebb, mint a bámészkodók által megvizsgált r_2 sugarú kör. Az ABC bámészkodói rulettkerék-szelekcióval [36, 25] kiválasztanak n_{ob} számú nektárforrást, és egy-egy bámészkodó megvizsgál csak egyetlen egyet (lásd 5. algoritmus 11–18 sora). A rulettkerék-szelekció biztosítja, hogy a ${}^{(G)}\mathcal{P}_{ob}$ populációba olyan egyedek (források) kerüljenek be, melyek környezetében nagy va-

lószínűséggel van jobb, de lehetőséget biztosít a rosszabbak bekerülésére azt feltételezve, hátha a környezetében van jobb.

StdBA felderítő méhei minden iterációs lépésben a ${}^{(G)}\mathcal{P}$ populáció n_{sb} darabszámú legrosszabb fitnesz értékű egyedei, melyek teljesen véletlenszerűen mozognak a keresési térben (lásd 4. algoritmus 13–14 sora). Az ABC minden egyedhez hozzárendel egy élethosszat. Ha egy méh élethossza meghalad egy küszöbértéket, más szavakkal, ha egy méh egy adott iterációs lépésszám után se talál az általa ismert forrásnál jobbat, akkor felderítő méhhé válik. Az StdBA-hoz hasonlóan a felderítés ABC esetén is teljesen véletlenszerű (lásd 5. algoritmus 19–21 sora).

2.1.6. Részecskeraj optimalizáció

A részecskeraj optimalizáció (PSO) [18, 101] az egyik legrégebbi rajintelligenciák közé sorolható evolúciós módszer. Napjainkban is népszerűen alkalmazott technika, mivel nagyon egyszerű felépítésű, és könnyen implementálható.

A keresési térben ${}^{(G)}\mathcal{P}$ populáció minden egyes ${}^{(G)}\mathbf{x}_i$ egyede (pontja) csak rá jellemző ${}^{(G)}\mathbf{v}_i$ sebességgel mozog így hozva létre iterációs lépésről iterációs lépésre a ${}^{(G+1)}\mathcal{P}$ populáció egyedeit.

$$ {}^{(G+1)}\mathbf{x}_i = {}^{(G)}\mathbf{x}_i + {}^{(G)}\mathbf{v}_i \,, \tag{2.24} $$

$$ {}^{(G)}\mathbf{v}_i = \chi {}^{(G)}\mathbf{x}_i + C_1 \mathcal{U}\left(0{,}1\right)\left({}^{(G)}\mathbf{x}_{best} - {}^{(G)}\mathbf{x}_i\right) + C_2 \mathcal{U}\left(0{,}1\right)\left({}^{(G)}\mathbf{x}_{gbest} - {}^{(G)}\mathbf{x}_i\right) \,, \tag{2.25} $$

ahol ${}^{(G)}\mathbf{x}_{best}$ a ${}^{(G)}\mathcal{P}$ populáció legfittebb egyede, ${}^{(G)}\mathbf{x}_{gbest}$ a G. iterációs lépésig megtalált legfittebb egyed, χ, C_1, C_2 konstansok az alábbiak szerint

$$ \Phi = \Phi_1 + \Phi_2 \,, \tag{2.26} $$

$$ \chi = \frac{2}{\Phi - 2 + \sqrt{\Phi^2 - 4\Phi}} \,, \tag{2.27} $$

$$ C_1 = \chi \Phi_1 \,, \quad C_2 = \chi \Phi_2 \,, \tag{2.28} $$

ahol Φ_1, Φ_2 konstansok és $\Phi_1 > 1$, $\Phi_2 > 1$.

2.1.7. Szentjánosbogár algoritmus

A szentjánosbogár algoritmus (FA) biológiai viselkedés ösztönözte módszer, melyet a [119] irodalomban Xin-She Yang mutatott be 2009-ben. A szentjánosbogarak kémiai úton előállított fényt bocsájtanak ki a sötétben. A tényleges funkciója még mind a mai napig vitatott téma, de a legelterjedtebb elméletek szerint fontos szerepet játszik a zsákmányszerzésben és a társegyedek vonzásában a szaporodás során. A bioluminescens alapú kommunikáció matematikai formalizmusának felírásához az alábbi egyszerűsítések alkalmazhatóak:

- minden szentjánosbogár egynemű, ezért minden egyed a fényességétől függően vonzhatja a másikat,
- a vonzerő arányos a fényességgel és a két bogár közötti távolsággal,
- mindig a kevésbé fényes vonzódik a fényesebbhez,
- ha két bogár fényere azonos, mindkét bogár véletlenszerűen mozog.

Algoritmus 6: Szentjánosbogár algoritmus pszeudokódja

```
(0)P populáció (0)x_i egyedeinek inicializálása
szükséges konstansok beállítása: α_damp, 0α_l, β_l, γ_l, m
while kilépési feltétel hamis do
    for j ∈ (G)P do
        for i ∈ (G)P do
            if Ĩ_j(r_ij) < Ĩ_i then
                j. egyed mozog i. egyed irányába (2.32) egyenlet szerint
                új pozíció alapján j. egyed fitneszértékének frissítése
    (G+1)α_l frissítése (2.33) egyenlet alapján
    (G)P populáció egyedeinek csökkenő sorba rendezése
    (G)P populációból legjobb fitnesz értékű egyed kiválasztása
utófeldolgozás, eredmények vizualizációja
```

A szentjánosbogarak $\tilde{I}$ fényessége arányos a fitnesz értékkel

$$\tilde{I}\left({}^{(G)}\mathbf{x}_i\right) \propto \mathcal{F}\left({}^{(G)}\mathbf{x}_i\right) , \tag{2.29}$$

sok esetben a gyakorlatban a két érték megegyezik. Az i. bogár szemszögéből nézve egy j. bogár fényessége közelíthető az alábbiak szerint

$$\tilde{I}_j(r_{ij}) = \tilde{I}\left({}^{(G)}\mathbf{x}_j\right) e^{-\gamma_l r_{ij}} , \text{ vagy} \quad \tilde{I}_j(r_{ij}) = \frac{\tilde{I}\left({}^{(G)}\mathbf{x}_j\right)}{1+\gamma_l r_{ij}^2} , \tag{2.30}$$

ahol γ_l a fényelnyelődési tényező, r_{ij} a két bogár közötti távolság

$$r_{ij} = \left\| {}^{(G)}\mathbf{x}_i - {}^{(G)}\mathbf{x}_j \right\| = \sqrt{\sum_{k=1}^{n_D} {}^{(G)}x_{i,k}^2 - {}^{(G)}x_{j,k}^2} . \tag{2.31}$$

A kevésbé fényes egyed elmozdul a fényesebb egyed felé

$${}^{(G+1)}\mathbf{x}_j = {}^{(G+1)}\mathbf{x}_j + \beta_l e^{-\gamma_l r_{ij}^m} \left({}^{(G+1)}\mathbf{x}_i - {}^{(G+1)}\mathbf{x}_j\right) + {}^{(G+1)}\boldsymbol{\alpha}_l \mathcal{U}(0,1) , \tag{2.32}$$

ahol β_l a vonzódási konstans $r_{ij} = 0$ távolság esetén, m a r_{ij} távolság kitevője, és ${}^{(G+1)}\boldsymbol{\alpha}_l$ a véletlen mozgás konstansait tartalmazó vektor, mely iterációs lépésenként csökken

$${}^{(G+1)}\alpha_l = \alpha_{damp}{}^{(G)}\boldsymbol{\alpha}_l , \tag{2.33}$$

ahol $\alpha_{damp} \in (0,1)$ konstans. A teljes kódot a 6. algoritmus szemlélteti, melyben jól kivehető, hogy egy iterációs lépés során egy egyed többször is mozoghat, és ami ennél még fontosabb, hogy a fitnesz érték is többször kerül meghatározásra.

2.1.8. Virágbeporzási algoritmus

A virágbeporzási algoritmust, [120]-ban publikálták először, melyet a növények reprodukciós folyamata ihletett. Két különböző rekombinációs módszert alkalmaz a már meglévő

példányokból újabb egyedek generálására. A globális beporzási modell során a külső fél (pl. madarak, rovarok, szél, stb.) által történő beporzást közelíti matematikai modellel

$$^{(G+1)}\mathbf{x}_{I,i} = {}^{(G)}\mathbf{x}_i + \mathcal{L}\left({}^{G}\mathbf{x}_i - {}^{G}\mathbf{x}_{best}\right) , \tag{2.34}$$

ahol $^{(G)}\mathbf{x}_i$ a (G). generáció i. egyede (a [120] és [123] irodalmak terminológiája szerint i. pollen), $^{(G)}\mathbf{x}_{best}$ a (G). generáció legjobb eredményét adó egyede, $\mathcal{L}$ pedig egy Lévy eloszlást követő $\mathcal{L} > 0$ véletlen szám. Lévy eloszlású véletlen számot lehet jó közelítéssel generálni a Mantegna algoritmus szerint [71], [122]

$$\mathcal{L} = \frac{u}{|v|^{\frac{1}{\lambda}}} , \qquad u \sim \mathcal{N}(0{,}1) , \qquad v \sim \mathcal{N}\left(0, \sigma^2\right) , \tag{2.35}$$

$$\sigma^2 = \frac{\Gamma(1+\lambda)}{\lambda\Gamma\left(\frac{1+\lambda}{2}\right)} \frac{\sin\left(\frac{\pi\lambda}{2}\right)^{\frac{1}{\lambda}}}{2^{\frac{\lambda-1}{2}}} , \tag{2.36}$$

ahol u és v egymástól független 0 középértékű, 1 illetve σ^2 szórás négyzetű normál eloszlást követő véletlen számok, λ egy (0,2] félig zárt intervallumon választott paraméter, $\Gamma(\lambda)$ λ helyen vett gamma függvény. Lokális, növényen belüli beporzás matematikai modellje

$$^{(G+1)}\mathbf{x}_{II,i} = {}^{(G)}\mathbf{x}_i + \epsilon\left({}^{(G)}\mathbf{x}_j - {}^{(G)}\mathbf{x}_k\right) \qquad j,k \in [1,n_p] \qquad i \neq j \neq k , \tag{2.37}$$

ahol $\epsilon = \mathcal{U}(0{,}1)$ egy [0,1) félig zárt intervallumon vett egyenletes eloszlású véletlen szám, j és k véletlen indexek. A (2.34) és (2.37) egyenletek által leírt rekombinációs technikák között egyedenként véletlenszerűen választ egy p_p valószínűségi változó alapján

$$^{(G+1)}\mathbf{x}_i = \begin{cases} {}^{(G+1)}\mathbf{x}_{I,i} & \text{ha } p_p \leq \mathcal{U}(0{,}1) \\ {}^{(G+1)}\mathbf{x}_{II,i} & \text{egyebként} \end{cases} . \tag{2.38}$$

2.2. Feltételes optimálás

Az evolúciós algoritmusok folytonos optimálási feladatok megoldására vannak kifejlesztve, kivétel néhány diszkrét feladatot megoldó eljárás [27, 89, 111]. A mérnöki gyakorlatban előforduló feladatok gyakrabban feltételes optimálási, keresési problémák

$$\begin{array}{lll} & & \mathbf{x} = (x_1, x_2, x_3, \ldots, x_i, \ldots, x_D) \\ \min & f(\mathbf{x}) & \mathbf{x} \in \mathcal{S}^D, \quad x_i \in [x_{L,i}; x_{U,i}] \\ \text{ha} & g_j(\mathbf{x}) < 1 & 1 \leq j \leq r \\ & h_k = 0 & r+1 \leq k \leq q \end{array} , \tag{2.39}$$

ahol az x_i egy eleme az $\mathbf{x}$ vektornak, $x_{L,i}$ és $x_{U,i}$ az alsó illetve felső határok, $g_j(\mathbf{x})$ az egyenlőtlenségi feltétel, $h_k(\mathbf{x})$ az egyenlőségi feltétel, r és q az egyenlőtlenségi és az egyenlőségi feltételek száma. A feltételes optimálást folytonossá kell tenni

$$\mathcal{F}(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{r} p_j(g_j(\mathbf{x})) + \sum_{k=r+1}^{q} p_k(h_k(\mathbf{x})) , \tag{2.40}$$

ahol $p_j()$ és $p_k()$ a büntetőfüggvények.

A büntetőfüggvények két nagy csoportja ismert, a belső és a külső büntető-függvény [49, 54]. Belső büntetőfüggvény esetén minden közbenső megoldás a feltételt kielégítő lehetséges megoldás. Előnye, hogy minden iterációs lépésben van jó egyed a populációban.

Hátránya, hogy már a populáció inicializálásakor is ismerni kell legalább egy lehetséges megoldást, továbbá nehezen automatizálhatóak vele a számítások. Ezért a belső büntetőfüggvényekkel az értekezés a továbbiakban nem foglakozik.

A külső büntetőfüggvények a feltételek megsértéséhez kapcsolódnak, értékük annál nagyobb, minél jobban megsérti az egyed az adott feltételt. Előnyük többek között, hogy jól automatizálható számításokat lehet velük készíteni, nincs szükség jó megoldásokra az optimálás elején. Hátrányuk, hogy nem garantált az optimálás végén a jó megoldás, és ebből következően nem feltétlen lehet bármikor megállítani az iterációt [49]. A külső büntetőfüggvények általános felépítése [127]

$$p_j(\mathbf{x}) = \begin{cases} 0 & \text{ha } g_j(\mathbf{x}) \leq 1 \\ \tilde{r}_j g_j(\mathbf{x}) & \text{egyébként} \end{cases}, \qquad p_k(\mathbf{x}) = \begin{cases} 0 & \text{ha } |h_k(\mathbf{x})| - \varepsilon \leq 0 \\ \tilde{r}_k |h_k(\mathbf{x}) - \varepsilon| & \text{egyébként} \end{cases}, \tag{2.41}$$

ahol ε a [41] cikk által javasolt hibahatár, $\tilde{r}_j$ és $\tilde{r}_k$ büntetőparaméterek. A büntetőparaméterek megválasztásánál törekedni kell

$$\tilde{r}_j,\ \tilde{r}_k \gg \min f(\mathbf{x}) \tag{2.42}$$

egyenlőtlenség kielégítésére. Szakirodalmi ajánlások alapján a büntetőparaméterek a szokásosan az alábbiak szerint választhatóak:

- *Konstans*: a generációszámtól független a teljes iterációs ciklus alatt állandó. Egyszerű és könnyen implementálható. Elfajult esetben, ha az implementációs rendszer (programozási nyelv) engedi $\tilde{r}_j = \tilde{r}_k = \infty$, akkor ezt a büntetőparamétert *death penalty*-nak szokás nevezni [7, 74]. A végtelen nagy paraméterek alkalmazása mérnöki feladatoknál nem javasolt, mert ha a kezdő populáció nem tartalmaz lehetséges megoldást, nincs lehetőség az egyedek rangsorolására. A külső büntetőfüggvény elveszti az egyik jól kihasználható előnyét.

- *Statikus büntetések* [41]: a feltételek megsértésének mértéke diszkrét tartományokra van bontva, és ezen tartományok alapján változik kiértékelésenként a büntetőparaméter értéke,

$$\tilde{r}_j(\mathbf{x}) = \begin{cases} \tilde{r}_{j,1} & \text{ha } g_j(\mathbf{x}) \in (1, \varepsilon_{j,1}] \\ \tilde{r}_{j,2} & \text{ha } g_j(\mathbf{x}) \in (\varepsilon_{j,1}, \varepsilon_{j,2}] \\ \tilde{r}_{j,3} & \text{ha } g_j(\mathbf{x}) \in (\varepsilon_{j,3}, \varepsilon_{j,3}] \\ \vdots & \\ \tilde{r}_{j,n-1} & \text{ha } g_j(\mathbf{x}) \in (\varepsilon_{j,n-2}, \varepsilon_{j,n-1}] \\ \tilde{r}_{j,n} & \text{ha } g_j(\mathbf{x}) \in (\varepsilon_{j,n-1}, \infty] \end{cases} \tag{2.43}$$

$$\tilde{r}_k(\mathbf{x}) = \begin{cases} \tilde{r}_{k,1} & \text{ha } |h_k(\mathbf{x})| - \varepsilon \in (0, \varepsilon_{k,1}] \\ \tilde{r}_{k,2} & \text{ha } |h_k(\mathbf{x})| - \varepsilon \in (\varepsilon_{k,1}, \varepsilon_{k,1}] \\ \tilde{r}_{k,3} & \text{ha } |h_k(\mathbf{x})| - \varepsilon \in (\varepsilon_{k,1}, \varepsilon_{k,1}] \\ \vdots & \\ \tilde{r}_{k,m-1} & \text{ha } |h_k(\mathbf{x})| - \varepsilon \in (\varepsilon_{k,m-2}, \varepsilon_{k,m-1}] \\ \tilde{r}_{k,m} & \text{ha } |h_k(\mathbf{x})| - \varepsilon \in (\varepsilon_{k,m-1}, \infty] \end{cases} \tag{2.44}$$

$$\varepsilon_{j,1} < \varepsilon_{j,2} < \varepsilon_{j,3} < \ldots < \varepsilon_{j,n-2} < \varepsilon_{j,n-1} < \infty \tag{2.45}$$

$$\varepsilon_{k,1} < \varepsilon_{k,2} < \varepsilon_{k,3} < \ldots < \varepsilon_{k,m-2} < \varepsilon_{k,m-1} < \infty\,, \tag{2.46}$$

ahol n, m a tartományok száma; $\tilde{r}_{j,1}, \tilde{r}_{j,2}, \ldots, \tilde{r}_{j,n}$ és $\tilde{r}_{k,1}, \tilde{r}_{k,2}, \ldots, \tilde{r}_{k,m}$ a tartományonkénti büntetőparaméterek; $\varepsilon_{j,1}, \varepsilon_{j,2}, \ldots, \varepsilon_{j,n-1}$ és $\varepsilon_{k,1}, \varepsilon_{k,2}, \ldots, \varepsilon_{k,m-1}$ a tartomány határok.

– *Dinamikus büntetés* [52]: az $\tilde{r}_j$ és $\tilde{r}_k$ értéke a generációk számával arányosan nő illetve csökken

$$\tilde{r}_j = (C_j \cdot G)^{\alpha_j} + \beta_j\,, \qquad \tilde{r}_k = (C_k \cdot G)^{\alpha_k} + \beta_k\,, \tag{2.47}$$

ahol G a generáció száma, C_j, C_k, α_j, α_k, β_j és β_k a felhasználó által választott konstansok. A [60] egy teljesen más megközelítést javasol a dinamikus büntetőparaméter előállítására

$$\tilde{r}_j = \left(\frac{G}{G_t}\right)^2 W_j \Phi_j(g(\mathbf{x})) + \beta_j\,, \qquad \tilde{r}_k = \left(\frac{G}{G_t}\right)^2 W_k \Phi_k(|h(\mathbf{x})| - \varepsilon) + \beta_k\,, \tag{2.48}$$

ahol G_t a teljes generáció szám, W_j és W_k a választott súly paraméterek, $\Phi_j()$ és $\Phi_k()$ a feltétel megsértésének mértékétől függő függvény és β_j és β_k szabadon választott konstansok. A paraméter ily módú előállítása garantálja, hogy az iteráció előrehaladtával a büntetőparaméter nő, és lehetőséget biztosít a feltételek súlyozására, megkülönböztetésére.

– *Adaptív büntetés*: a büntetőparaméter értéke generációnként frissül, figyelembe véve a populáció minőségét, különös tekintettel a feltételek megsértésére [37]

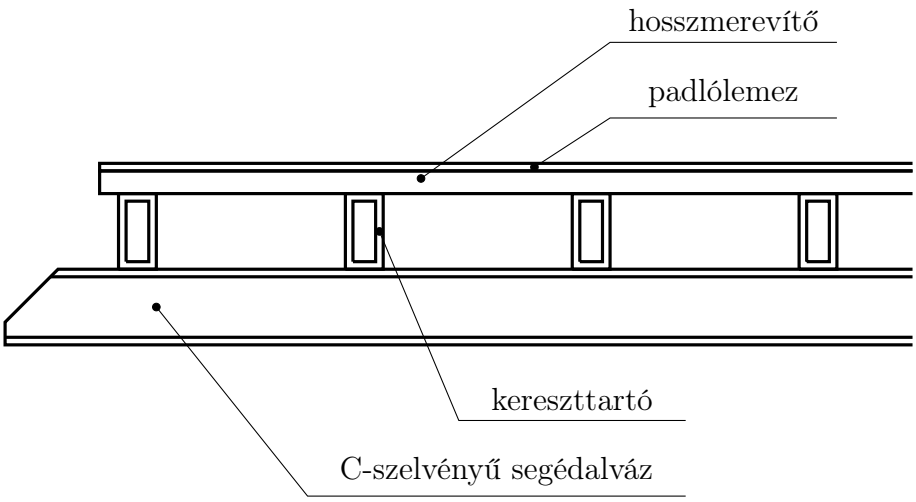
$${}^{(G+1)}\tilde{r}_j = \begin{cases} \frac{1}{\tilde{r}_{j,1}}\,{}^{(G)}r_j & \text{ha } {}^{(G)}n_{g,a} < n_{g,lim} \\ \tilde{r}_{j,2}\,{}^{(G)}r_j & \text{ha } {}^{(G)}n_{g,a} > n_{g,lim} \\ {}^{(G)}r_j & \text{ha } {}^{(G)}n_{g,a} = n_{g,lim} \end{cases} \tag{2.49}$$

$${}^{(G+1)}\tilde{r}_k = \begin{cases} \frac{1}{\tilde{r}_{k,1}}\,{}^{(G)}r_k & \text{ha } {}^{(G)}n_{h,a} < n_{h,lim} \\ \tilde{r}_{k,2}\,{}^{(G)}r_k & \text{ha } {}^{(G)}n_{h,a} > n_{h,lim} \\ {}^{(G)}r_k & \text{ha } {}^{(G)}n_{h,a} = n_{h,lim} \end{cases}, \tag{2.50}$$

ahol $\tilde{r}_{j,1}$, $\tilde{r}_{j,2}$, $\tilde{r}_{k,1}$, $\tilde{r}_{k,2}$ választott konstansok, ${}^{(G)}n_{g,a}$, ${}^{(G)}n_{h,a}$ az aktuális populáció egyedei által összesen megsértett egyenlőtlenségi és egyenlőségi feltételek száma, $n_{g,lim}$, $n_{h,lim}$ a populáció minősítéséhez szükséges megsérthető egyenlőtlenségi és egyenlőségi feltételek száma (független attól, hogy az adott egyed egy lehetséges megoldás, vagy nem).

2.3. Teherautó platós kereszttartóinak optimálása

Jelen fejezetben egy háromrétegű teherautó platós kereszttartóinak evolúciós technikákkal történő optimálásához szükséges egyenletek kerülnek bemutatásra. A platós rétegrendjét szemlélteti a 2.1. ábra. A jármű alvázához a két hosszirányú segédalvázon keresztül kapcsolódik a platós. Először a segédalváz tetejére fektetik a keresztirányú kereszttartókat. A padlólemez $h_c = 34\,\text{mm}$ magas RHS hosszmerevítőkkel kapcsolódik kereszttartókhoz. Eredetileg a kereszttartók RHS profilból készülnek, míg a segédalváz C-szelvényű. Az eredetileg RHS szelvény helyett érdemesebb I-szelvényeket alkalmazni kereszttartónak, mivel szerszámozási költségük alacsonyabb [50].



2.1. ábra. Háromrétegű teherautó plató szerkezete

A segédalváz és a kereszttartók [32] szerinti En AW-6005A alumíniumból készülnek, mely folyáshatára $f_y = 215\,\mathrm{MPa}$. A padlólemez anyag pedig En AW-5052. A szerkezeti elemek hegesztett kötésekkel kapcsolódnak egymáshoz.

Jelen esetben azzal a feltételezéssel élek, hogy a teljes teherviselő szerkezet maga a plató. A felépítmény teherviselő képessége – ha egyáltalán van szerepe a teherviselésben – elhanyagolásra kerül.

2.3.1. Terhelési állapot

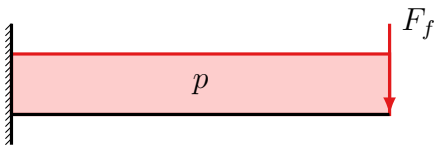
A vizsgált terhelési állapotban a terhelés a felépítmény súlyából és a hasznos teherből adódik. Jelen esetben a plató maximális terhelhetősége $F_p = 127{,}5\,\mathrm{kN}$. Zárt, merev falú felépítményt feltételezve, ennek a súlyából (úgy mint oldalfalak, tető, ajtók stb. súlyának összege) egy kereszttartóra átadódó terhelés $F_f = 1946\,\mathrm{N}$.

A kereszttartók segédvázon túlnyúló részét a 2.2. ábra szerint befalazott tartóként lehet modellezni. A plató maximális terhelhetőségének kihasználása mellett feltételezem, hogy a terhelés egyenletesen oszlik el a teljes felületen. Konzolos szakasz hossza a plató szélességének körülbelül egyharmada. Erre a felületre jutó megoszló terhelés

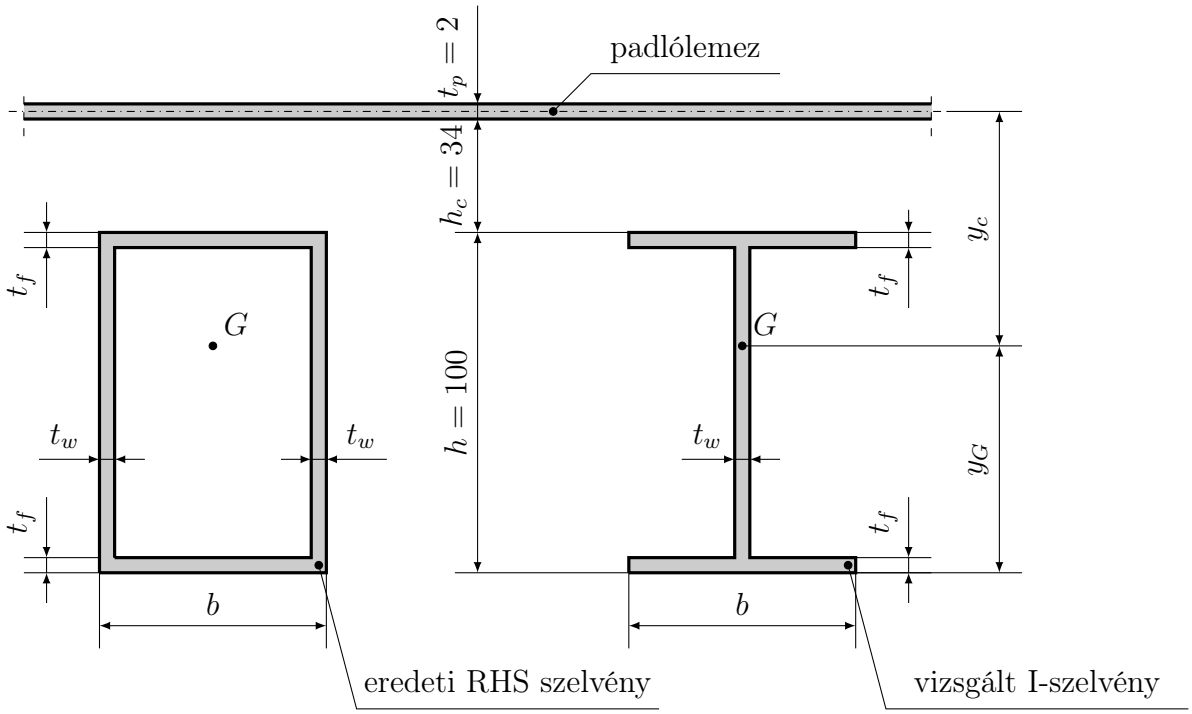
$$p = \frac{F_p}{3L_cL_p}\,, \tag{2.51}$$

ahol a $L_c = 720\,\mathrm{mm}$ a kereszttartó túlnyúló részének hossza, $L_p \approx 6000\,\mathrm{mm}$ pedig a teljes plató hossza. Egy kereszttartóra ható megoszló terhelés pedig

$$p_c = \frac{pL_p}{n_c - 1} = \frac{F_p}{3L_c\,(n_c - 1)}\,, \tag{2.52}$$



2.2. ábra. Kereszttartók túlnyúló része, mint befalazott tartó



2.3. ábra. Kereszttartó és közvetlen környezetének méretei

ahol n_c a kereszttartók száma. Terhelésekből adódó hajlítónyomaték maximuma

$$M_h = \frac{p_c L_c^2}{2} + F_f L_c = \frac{F_p L_c}{6\,(n_c - 1)} + F_f L_c\,, \tag{2.53}$$

a mértékadó nyíróerő pedig

$$Q = \frac{F_p}{3\,(n_c - 1)} + F_f\,. \tag{2.54}$$

2.3.2. Optimálási feladat

Az optimálás célja a teherautó plató súlyának csökkentése. Az optimálandó célfüggvény csak a kereszttartó tömegét tartalmazza, mert a környezetében lévő egyéb szerkezeti elemek (hosszmerevítő, padlólemez, segédváz) nem kerülnek optimálásra

$$m_c = \rho_{Al} A_c L_{ct} n_c\,, \tag{2.55}$$

ahol $\rho_{Al} = 2{,}7 \cdot 10^{-6}\,\mathrm{kg/mm^3}$ az alumínium sűrűsége, $L_{ct} \approx 2440\,\mathrm{mm}$ a kereszttartó teljes hossza, A_c a 2.3. ábra jelöléseivel I-szelvényű tartó keresztmetszeti területe

$$A_c = (h - 2t_f)\,t_w + 2bt_f\,, \tag{2.56}$$

ahol h a gerinclemez magassága, t_w a gerinclemez vastagsága, b az övlemez szélessége és t_f az övlemez vastagsága. Az optimált tartónak egy az egyben be kell férnie az eredetileg RHS szelvényű elem helyére, ezért a gerinclemez magassága nem része az ismeretlenekből épített vektornak, és nem változik az iterációs lépések során.

A hajlításra és nyírásra igénybe vett keresztmetszet egyrészt a tartó keresztmetszetéből áll, másrészt az együtt dolgozó lemezszélességgel figyelembe vett padlólemezből. A

padlólemez keresztmetszeti területe kifejezhető a t_p lemezvastagsággal, mivel az együtt dolgozó lemezszélesség kifejezhető a horpadási feltételekből [49]

$$A_p = 50t_p^2\,. \tag{2.57}$$

Ezek alapján a vizsgált teljes keresztmetszet területe

$$A = A_c + A_p\,, \tag{2.58}$$

és másodrendű nyomatéka

$$I_x = \frac{h^3 t_w}{12} + \frac{bt_f h^2}{2} + A_c\left(y_c - \frac{h}{2}\right)^2 + A_p y_G^2\,, \tag{2.59}$$

ahol y_G a súlyponti tengely magassága a segédváztól, y_c a padlólemez középvonalának magassága a súlyponti tengelytől

$$y_G = \frac{0{,}5A_p h + A_c(h + h_c + 0{,}5t_p)}{A}\,, \tag{2.60}$$

$$y_c = h + h_c + \frac{t_p}{2} - y_G\,. \tag{2.61}$$

A terhelésből ébredő normál feszültség

$$\sigma_h = \frac{M_h}{I_x}\max\left(y_G, y_c\right)\,, \tag{2.62}$$

és nyírófeszültség közelítőleg [49, 31]

$$\tau_{yz} = \frac{Q}{(h - 2t_f)\,t_w}\,. \tag{2.63}$$

A kereszttartók mind a segédalvázhoz, mind hosszmerevítőkhöz hegesztett kötéseken keresztül kapcsolódnak. A varratokat jelen esetben fáradásra kell méretezni. [40] ajánlásai alapján a szerkezetnél alkalmazott varratok határfeszültsége alumínium alapanyag esetén $\Delta\sigma_C = 28\,\mathrm{MPa}$ és $\Delta\tau_C = 28\,\mathrm{MPa}$. Ezt átszámolva $N = 2 \cdot 10^5$ ciklusszámra az alkalmazandó határfeszültség hajlításra

$$\log\Delta\sigma_N = \frac{1}{3}\log\frac{2\cdot 10^6}{N} + \log\Delta\sigma_C \qquad \Delta\sigma_N = 60{,}3\,\mathrm{MPa}\,, \tag{2.64}$$

és nyírásra

$$\log\Delta\tau_N = \frac{1}{5}\log\frac{2\cdot 10^6}{N} + \log\Delta\tau_C \qquad \Delta\tau_N = 44{,}4\,\mathrm{MPa}\,. \tag{2.65}$$

A fáradásra vonatkozó követelmények és a (2.62), (2.63) egyenletek alapján felírható. Két alkalmazandó egyenlőtlenségi feltétele az optimalizálásnak

$$g_1(\mathbf{x}) = \frac{\gamma_f \sigma_h}{\Delta\sigma_N} \le 1\,, \tag{2.66}$$

$$g_2(\mathbf{x}) = \frac{\gamma_f \sqrt{3}\tau_{xy}}{\Delta\tau_N} \le 1\,, \tag{2.67}$$

ahol $\gamma_f = 1{,}25$ általános biztonsági tényező [31] szerint.

Gerinclemez helyi horpadását elkerülni célzó egyenlőtlenségi feltétel [20] alapján

$$g_3(\mathbf{x}) = \frac{\beta_g h}{22 t_w \varepsilon} \leq 1\,, \tag{2.68}$$

ahol

$$\beta_g = \begin{cases} 0{,}65 + 0{,}35 y_0/y_c & \text{ha } 0 \leq y_0/y_c \leq 1 \\ 0{,}65 + 0{,}30 y_0/y_c & \text{ha } -1 \leq y_0/y_c < 0 \end{cases}, \tag{2.69}$$

$$\varepsilon = \sqrt{\frac{250}{1{,}5\sigma_h}}\,, \tag{2.70}$$

$$y_0 = y_G - \frac{t_p}{2} - h_c\,. \tag{2.71}$$

Hasonlóan az előzőekhez az övlemez helyi horpadására vonatkozó feltétel

$$g_4(\mathbf{x}) = \frac{b}{7 t_f \varepsilon} \leq 1\,. \tag{2.72}$$

A (2.55), (2.66), (2.67), (2.68) és (2.72) egyenleteket összefűzve az optimálandó fitnesz függvény a kereszttartó optimálásához

$$\mathcal{F}(\mathbf{x}) = m_c + \sum_{i=1}^{4} p_i(g_i(\mathbf{x}))\,, \tag{2.73}$$

és a tervezési változókat tömörítő vektor (egyed)

$$\mathbf{x} = [b, t_f]\,, \tag{2.74}$$

ahol az alkalmazott $p_i(\ )$ statikus büntetőfüggvény

$$p_i(\mathbf{x}) = \begin{cases} 0 & \text{ha } g_i(\mathbf{x}) \leq 1 \\ 10^6 \cdot g_i(\mathbf{x}) & \text{egyébként} \end{cases}. \tag{2.75}$$

2.4. Futódaru-főtartójának optimálási modellje

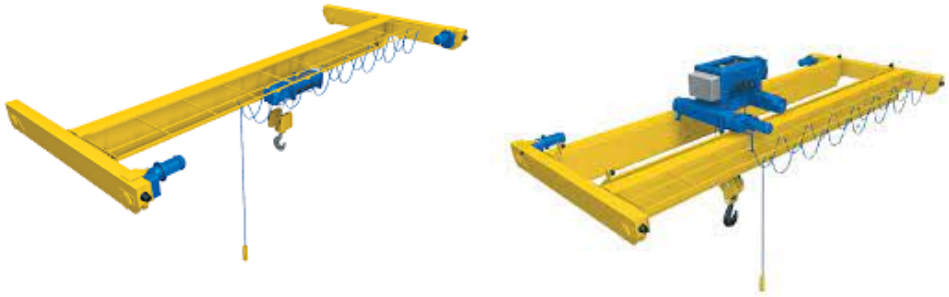
2.4.1. Tárgyalt főtartó adatai

A felülfutós futódaruk készülhetnek egy (lásd 2.4a. ábra) illetve két főtartóval (lásd 2.4b. ábra). Jelen esetben alkalmazott két főtartós futódaru, egy főtartóját szemlélteti a 2.5a. ábra. Az azonos méretekkel jellemezhető I-szelvényhez képest nagyobb csavarási merevséggel rendelkező hegesztett szekrényszelvényből készül a főtartó. Teljes hosszában egyenlő osztással elhelyezett, összesen 11db diafragma merevíti. A tartó közepén 5db *a* jelű (2.5b. ábra) könnyített diafragma van beépítve, míg a két végén *b* jelű (2.5c. ábra) teljes méretű diafragma. A sín elhelyezhető a szelvény közepén vagy az egyik gerinclemez felett. Most a belső gerinclemez felé lett elhelyezve.

A szekrényszelvény jellemző mennyiségei a 2.5b. és a 2.5c. ábra jelöléseivel:

– keresztmetszeti terület

$$A = h t_w + 2 b t_f\,, \tag{2.76}$$



(a) egy főtartós változat (b) két főtartós változat

2.4. ábra. Felülfutós futódaruk változatai

– másodrendű nyomatékok

$$I_x \cong \frac{h^3 t_w}{12} + 2bt_f \left(\frac{h}{2}\right)^2 , \text{ és} \qquad I_y \cong \frac{b^3 t_f}{6} + ht_w \left(\frac{b}{2}\right)^2 , \tag{2.77}$$

– keresztmetszeti tényezők [31] jelöléseit alkalmazva:

$$W_x \cong \frac{h^2 t_w}{6} + bht_f , \text{ és} \qquad W_y \cong \frac{b^2 t_f}{3} + \frac{ht_w b}{2} . \tag{2.78}$$

A főtartó statikai modellje egy kéttámaszú tartó, melyet a mértékadó esetben középen terhel kettő F erő. Az F erő hasznos P_h horogteherből és a macska súlyából származtatható $G_k = 45{,}25\,\text{kN}$

$$F = \frac{\Psi_d P_h + G_k}{4} , \tag{2.79}$$

ahol [19] szerint egy nagy igénybevételű műhelydaru dinamikus tényezője: $\Psi_d = 1{,}3$. Figyelembe kell még venni a szelvény p_s önsúlyát

$$p_s = 1{,}05\rho_0 A \qquad \rho_0 = 7{,}85 \cdot 10^{-5} \frac{\text{N}}{\text{mm}^3} , \tag{2.80}$$

továbbá a járda, és $h_s = 70\,\text{mm}$ magas sín $p_r = 1900\,\text{N/m}$ önsúlyát.

A veszélyes keresztmetszet mértékadó pontjában a σ_z főfeszültség az x tengely menti M_{hx} hajlításból és y tengely menti M_{hy} hajlításból származik. M_{hx} a statikus erőket tartalmazza, míg a M_{hy} a [19] szerint a gyorsításkor és a lassításkor fellépő dinamikus erőket,

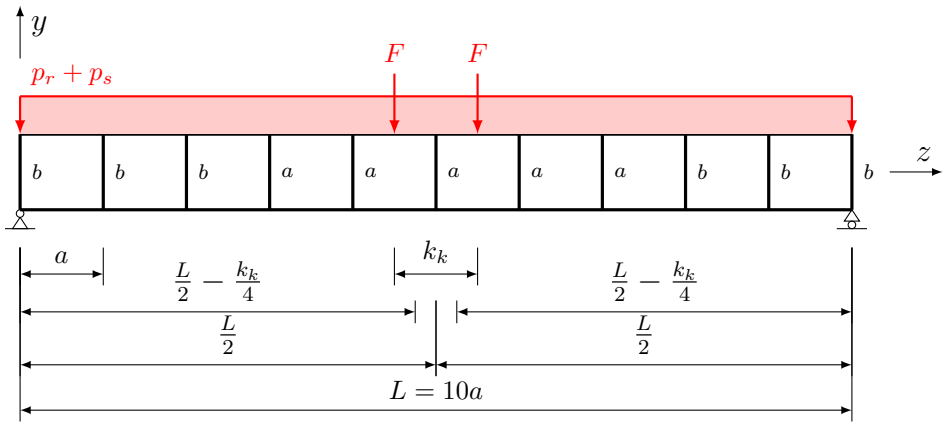
$$\sigma_1 = \frac{M_{hx}}{W_x} , \text{ ahol} \qquad M_{hx} = (1{,}05\rho_0 A + p_r + p_s)\frac{L^2}{8} + \frac{F}{2L}\left(L - \frac{k_k}{2}\right)^2 \tag{2.81}$$

$$\sigma_2 = \frac{M_{hy}}{W_y} , \text{ ahol} \qquad M_{hy} = 0{,}3 \cdot 0{,}5 \left[(1{,}05\rho_0 A) + \frac{G_k}{8L}\left(L - \frac{k_k}{2}\right)^2\right] \tag{2.82}$$

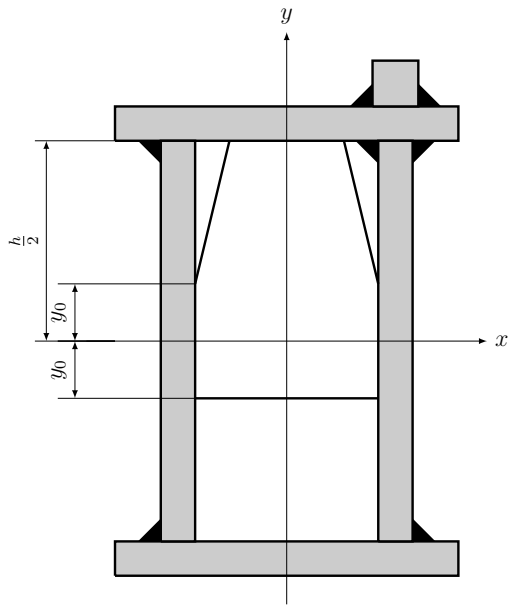
$$\sigma_z = \sigma_1 + \sigma_2 . \tag{2.83}$$

A keréknyomásból származó σ_y normál feszültség egy egyenlő szárú trapéz mentén oszlik szét (lásd 2.6. ábra), melynek alapon fekvő szögei 45°-osak

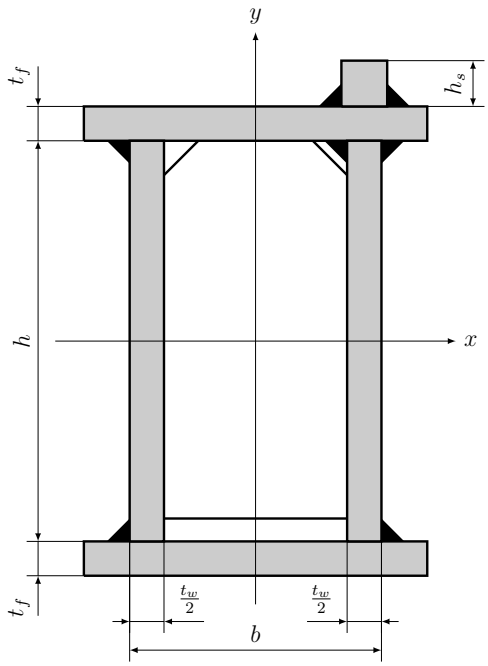
$$\sigma_y = \frac{2F}{ct_w} , \text{ ahol} \qquad c = 50 + 2\,(h_s + t_f) . \tag{2.84}$$



(a) főtartó vázlata



(b) keresztmetszet az a jelű diafragmáknál



(c) keresztmetszet a b jelű diafragmáknál

2.5. ábra. A futódaru főtartójának adatai és keresztmetszetei

A kerekeken átadó F erő támadáspontja nem megy keresztül a szelvény súlypontján ezért, csavarásból származó τ_t nyírófeszültséggel is kell számolni

$$\tau_t = \frac{2M_t}{2bht_w}\,, \text{ ahol} \qquad M_t = \frac{F}{2L}\left(L - \frac{k_k}{2}\right)\frac{b}{2} + \frac{(p_r + p_s)\,Lb}{4}\,. \tag{2.85}$$

A nyíróerőkből származó τ_V nyírófeszültség közelítőleg [49, 31]

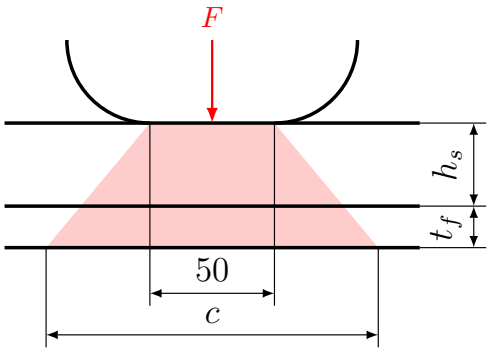
$$\tau_V = \frac{Q}{ht_w}\,, \tag{2.86}$$

ahol a Q nyíróerő

$$Q = (1{,}05\rho_0 A + p_r + p_s) + \frac{F}{2L}\left(L - \frac{k_k}{2}\right)\,. \tag{2.87}$$

A 2.5c ábra szerinti x tengely mentén a τ_{yz} nyírófeszültség

$$\tau_{yz} = \tau_V + \tau_t\,. \tag{2.88}$$



2.6. ábra. Keréknyomás eloszlása a gerinclemezen

2.4.2. Költségfüggvény

A hegesztett szekrényszelvényű tartó költségét általánosan a

$$K = k_m \rho V + k_w \sum_i T_i \tag{2.89}$$

alakban keressük. Vagyis áll egy anyag- és egy T_i idővel arányos gyártási költségből. A gyártási költséghez sok művelet tartozhat, mint például vágás, összeállítás és összefűzés, hegesztés, festés stb. Gyakorlati tapasztalatok alapján a legnagyobb hányadot az összeállítás, az összefűzés és a hegesztés teszi ki. Ezért jelen esetben is csak ez a két költségnem kerül figyelembevételre az anyagköltség mellett.

A T_a összeállítási és fűzési idő [49] közelítése alapján

$$T_a = C_a \Theta \sqrt{\kappa \rho V}\,, \tag{2.90}$$

ahol Θ a bonyolultsági tényező, κ az összehegesztendő, fűzendő elemek száma. A T_w hegesztés időszükségletének a meghatározásához több közelítő képlet is rendelkezésre áll. [57] szerinti közelítés egy varrat elkészítéséhez

$$T_w = 1{,}3 C_w a_w^n L_w\,, \tag{2.91}$$

ahol a C_w konstans és az a_w varrat mérethez tartozó n kitevő függ a hegesztési technológiától és a varrat helyzetétől. Javaslatok konkrét értékekre megtalálhatóak a [49] és az [57] irodalmakban.

A ténylegesen felmerülő és figyelembevett időszükségletek a gyártási sorrendnek megfelelően kerülnek megfogalmazásra CO_2 védőgázas ívhegesztéssel (GMAW-C).

A két gerinclemez összehegesztése 1500 mm hosszú acél lemezekből történik K tompa varratokkal $\Theta_1 = 2$ bonyolultsági tényezővel [53]

$$T_1 = \Theta_1 \sqrt{\kappa_1 \rho V_1} + 1{,}3 \cdot 0{,}152 \cdot 10^{-3} \left(\kappa_1 - 1\right) h \left(\frac{t_w}{2}\right)^{1{,}94}. \tag{2.92}$$

Ebben a munkafázisban jellemző térfogat

$$V_1 = \frac{L h t_w}{2}\,, \tag{2.93}$$

és a szerkezeti részek száma

$$\kappa_1 = \frac{L}{1500}\,. \tag{2.94}$$

A kapott eredményt nála nagyobb legkisebb egészre kell kerekíteni. A két övlemez gyártása ugyanúgy történik, mint a gerinclemezeké 1500 mm-es lemezekből [53]

$$T_2 = \Theta_1 \sqrt{\kappa_1 \rho V_2} + 1{,}3 \cdot 0{,}152 \cdot 10^{-3} \left(\kappa_1 - 1\right) b t_f^{1{,}94} \,, \tag{2.95}$$

ekkor a jellemző térfogat

$$V_2 = L b t_f \,. \tag{2.96}$$

Következő lépés a szekrényszelvény gerinclemezének, övlemezének, keresztdiafragmák összefűzése, összehegesztése, és a diafragmákat bekötő varratok utókezelése. Ekkor a térfogat

$$V_3 = L\left(h t_w + b t_f\right) + 6 b h t_s + 2{,}5 b t_s \left(h + 2 y_0\right), \qquad y_0 = \frac{h}{2 \cdot 1{,}6} \,, \tag{2.97}$$

ahol $t_s = 6\,\text{mm}$ a diafragmák lemezvastagsága. Az összeállított szerkezeti elemek száma $\kappa_3 = 14$, és a bonyolultsági tényező $\Theta_3 = 3$. A teljes időszükséglet négy részből tevődik össze, úgy mint: T_3 két gerinclemez és felső övlemezt összekötő varratok, T_4 sín alatti K-tompavarratok, T_5 diafragmák bekötései, és végül ez utóbbiak T_6 utókezeléséhez szükséges gyártási idők [53]

$$T_3 = \left(\Theta_3 \sqrt{\kappa_3 \rho V_3} + 3 \cdot 1{,}3 \cdot 0{,}3394 \cdot 10^{-3} a_w^2 L\right) \,, \tag{2.98}$$

$$T_4 = 2 \cdot 1{,}3 \cdot 0{,}1520 \cdot 10^{-3} a_w^{1{,}94} L \,, \tag{2.99}$$

$$T_5 = 1{,}3 \cdot 0{,}7889 \cdot 10^{-3} a_w^2 L_w \,, \text{ ahol} \qquad L_w = 2\left[6\left(b + 2h\right) + 5\left(b + 2 y_0\right)\right] \,, \tag{2.100}$$

$$T_6 = 10 b T_0 \,, \qquad T_0 = 0{,}0033 \, \frac{\text{min}}{\text{mm}} \,. \tag{2.101}$$

Utolsó munkafázis az alsó övlemez hozzáhegesztése kettő sarokvarrattal az eddig elkészült szekrényszelvényhez. Az így kapott teljes főtartó jellemző térfogata

$$V = L\left(h t_w + b t_f\right) + 6 b h t_s + 2{,}5 b t_s \left(h + 2 y_0\right) + b t_f L = V_3 + b t_f L \tag{2.102}$$

a hegesztési idő pedig [53]

$$T_7 = \left(\Theta_7 \sqrt{\kappa_7 \rho V} + 2 \cdot 1{,}3 \cdot 0{,}3394 \cdot 10^{-3} a_w^2 L\right) \,, \text{ ahol} \qquad \Theta_7 = 2,\ \kappa_7 = 2 \,. \tag{2.103}$$

A fent részletezett gyártási időket behelyettesítve a (2.89) egyenletbe, megkapjuk a vizsgált költséget

$$K = k_m \rho V + k_w \left(2 \sum_{i=1}^{2} T_i + \sum_{i=3}^{7} T_i\right) \,. \tag{2.104}$$

2.4.3. Optimálási feltételek

Mérnöki szempontból a megfelelő szerkezetnek az optimálás során számos feltételt ki kell elégíteni, mint szilárdsági, stabilitási, fáradási és egyéb szabványos feltételek. Ezekből a feltételekből értelmezhető a terhelés, és a megengedett maximális terhelés hányadosaként a keresztmetszeti kihasználtság, ami majd az optimálás tényleges egyenlőtlenségi feltételrendszerét adja.

Szilárdsági szempontból ellenőrzési feltételként definiálható, hogy az ébredő feszültségek egyenként és redukálva se haladhatják meg a f_y folyáshatár egy a [19], [31] és [78] szabványok szerint k_z, k_y és $k_{\tau 0}$ biztonsági tényezőkkel módosított értékét

$$g_1 = \frac{\sigma_z}{k_z f_y} \leq 1\,, \tag{2.105}$$

$$g_2 = \frac{\sigma_y}{k_y f_y} \leq 1\,, \tag{2.106}$$

$$g_3 = \frac{\sqrt{3}\tau_{yz}}{k_{\tau 0}} \leq 1\,, \tag{2.107}$$

ahol [31] szerint $k_z = k_y = k_{\tau 0} = 1$ biztonsági tényezők. A redukált feszültségre vonatkozó keresztmetszeti kihasználtság pedig az alábbi egyszerű formában írható fel

$$g_4 = \frac{\sqrt{\sigma_z^2 + \sigma_y^2 - \sigma_z\sigma_y + 3\tau_{yz}^2}}{f_y} \leq 1\,. \tag{2.108}$$

A vékonyfalú rudaknál az oldalakat alkotó lemezek stabilitásvesztése (lemez horpadás) esetén a [31] szerint együtt dolgozó lemezszélességgel kell számolni. Ez az eredeti lemezszélességnek egy csökkentett értéke. Könnyen belátható, hogy az optimális keresztmetszet megtalálásához egy olyan kihasználtsági feltételt kell bevezetni, ami még éppen elkerüli a lemezhorpadást. Így nem kell számolni együtt dolgozó lemezszélességgel. A gerinclemez horpadási feltételei a hajlítások, a keréknyomás, és végül a nyírás alapján [31]

$$g_5 = \frac{0{,}673 \cdot 28{,}42\varepsilon\sqrt{\tilde{k}_{\sigma x}}t_w}{2h} \leq 1\,, \tag{2.109}$$

$$g_6 = \frac{60{,}67\varepsilon t_w}{2h} \leq 1\,, \tag{2.110}$$

$$g_7 = \frac{31\varepsilon\sqrt{\tilde{k}_{\tau}}t_w}{2h}\,, \tag{2.111}$$

ahol ε, $\tilde{k}_{\sigma x}$, $\tilde{k}_{\tau}$ a [31] szerint kerültek felvételre

$$\varepsilon = \sqrt{\frac{235}{f_y}}\,, \tag{2.112}$$

$$\tilde{k}_{\sigma x} = 7{,}81 - 6{,}29\Psi_x + 9{,}81\Psi_x^2\,, \text{ ahol} \qquad \Psi_x = -\frac{\sigma_{hx} - \sigma_{hy}}{\sigma_{hx} + \sigma_{hy}}\,, \tag{2.113}$$

$$\tilde{k}_{\tau} = 5{,}34 + \frac{4h^2}{a^2} = 5{,}34 + \frac{400h^2}{L^2}\,. \tag{2.114}$$

Az övlemez horpadási feltételei a gerinclemezéhez hasonlóan

$$g_8 = \frac{0{,}673 \cdot 28{,}42\varepsilon\sqrt{\tilde{k}_{\sigma y}}t_f}{b}\,, \tag{2.115}$$

$$g_9 = \frac{31\varepsilon\sqrt{\tilde{k}_{\tau b}}t_f}{b} \leq 1\,, \tag{2.116}$$

ahol a $\tilde{k}_{\sigma y}$ és $\tilde{k}_{\tau b}$ tényezők ismét szabványi ajánlások

$$\tilde{k}_{\sigma y} = \frac{8{,}2}{1{,}05 + \Psi_y}\,, \text{ ahol} \qquad \Psi_y = \frac{\sigma_{hx} - \sigma_{hy}}{\sigma_{hx} + \sigma_{hy}}\,, \tag{2.117}$$

$$\tilde{k}_{\tau b} = 5{,}34 + \frac{4b^2}{a^2}\,. \tag{2.118}$$

Az időben változó terhelés miatt a szerkezetet fáradásra is szükséges ellenőrizni. Fáradás szempontjából elegendő csak a varratokat vizsgálni, azok közül is a darusín alatti nyakvarratot és a diafragmákat bekötő sarokvarratokat. A fáradási határfeszültség terhelési ciklusszám függvényében logaritmikusan változik [49]

$$\log \Delta\sigma_N = \begin{cases} \dfrac{1}{3}\log\dfrac{2\cdot 10^6}{N} + \log\Delta\sigma_C & \text{ha } N \le 5\cdot 10^6 \\ \dfrac{1}{5}\log\dfrac{5\cdot 10^6}{N} + \log\Delta\sigma_D & \text{ha } 5\cdot 10^6 < N \le 10^8 \\ \log\left(\sqrt[5]{0{,}05}\Delta\sigma_D\right) & \text{ha } 10^8 < N \end{cases}, \tag{2.119}$$

ahol

$$\log \Delta\sigma_D = \log\left(\sqrt[3]{\frac{2}{5}}\Delta\sigma_C\right)\,. \tag{2.120}$$

A nyírófeszültség esetén is hasonlóan lehet számítani a határfeszültséget, itt csak egy töréspontja lesz a görbének [49]

$$\log \Delta\tau_N = \begin{cases} \dfrac{1}{5}\log\dfrac{2\cdot 10^6}{N} + \log\Delta\tau_C & \text{ha } N \le 10^8 \\ \log\left(\sqrt[5]{0{,}02}\Delta\tau_C\right) & \text{ha } 10^8 < N \end{cases}. \tag{2.121}$$

Természetesen mind a $\Delta\sigma_N$, mind a $\Delta\tau_N$ határfeszültségnek felső korlátja az anyag f_y folyáshatára. A határfeszültséget felhasználva [78] szerint a K nyakvarrat fáradására a feltétel

$$g_{10} = \left(\frac{\gamma_f \sqrt[3]{s_3}\sigma_z}{\Delta\sigma_N}\right)^3 + \left(\frac{\gamma_f \sqrt[3]{s_3}\sigma_y}{\Delta\sigma_N}\right)^3 + \left(\frac{\gamma_f \sqrt[3]{s_3}\tau_{yz}}{\Delta\tau_N}\right)^3 \le 1\,, \tag{2.122}$$

ahol szabvány szerint $s_3 = 1$ a spektrumtényező, $\gamma_f = 1{,}25$ a fáradási biztonsági tényező. A diafragmákat bekötő sarokvarratokra vonatkozó feltétel HiFT nagyfrekvenciás ütőkezelést feltételezve

$$g_{11} = \frac{\gamma_f \sigma_z}{\alpha_P \Delta\sigma_N} \le 1\,, \tag{2.123}$$

ahol $\alpha_P = 1{,}6$ az utókezelésre vonatkozó tényező.

Az [78] szabvány egy lehajlási feltételt is meghatároz, melyet ki kell elégíteni. A w lehajlás közelíthető

$$w = \frac{F\left(L-k_k\right)\left(3L^2 - \left(L-k_k\right)^2\right)}{96EI_x} + \frac{5\left(1{,}05\rho_0 A + p_r + p_s\right)L^4}{384EI_x} \tag{2.124}$$

alakban. Maga a feltétel pedig

$$g_{12} = \frac{600w}{L} \le 1\,. \tag{2.125}$$

A (2.105) - (2.125) egyenlőtlenségi feltételeket a közvetlenül az evolúciós algoritmusok nem tudják kezelni, mivel megszűnik a keresési tér folytonossága. Külső büntetőfüggvény alkalmazásával a keresési tér újra folytonossá tehető

$$p_i(\mathbf{x}) = \begin{cases} 0 & \text{ha } g_i \leq 1 \\ Rg_i & \text{ha } 1 < g_i \end{cases}, \tag{2.126}$$

ahol R egy kellően nagy konstans (büntetőparaméter), akár végtelen is. Végtelent alkalmazva a feltételek megsértésének mértéke között nem lehet különbséget tenni, és már az első populációban is kell lennie legalább egy lehetséges megoldásnak. Ez nem mindig kivitelezhető, célszerű R értékét olyan nagyra választani, hogy az optimálandó függvény várható értékénél 1 - 2 nagyságrenddel legyen nagyobb. Túl nagy büntetőparaméter könnyen túlcsordulást eredményezhet. Jelen esetben $R = 10^6$, mert a költség becsült értéke 10^4 - 10^5 tartományban van.

A (2.104) és (2.126) egyenleteket felhasználva a minimalizálandó fitneszfüggvény

$$\mathcal{F}(\mathbf{x}) = K + \sum_{i=1}^{11} p_i(g_i(\mathbf{x})), \tag{2.127}$$

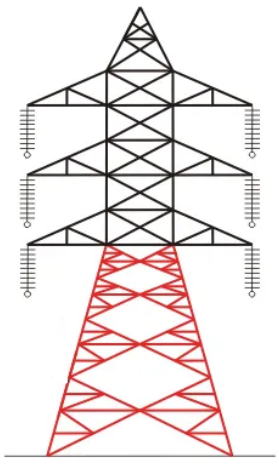
és az ismeretlenek (tervezési változókat) tömörítő vektor (egyed)

$$\mathbf{x} = [h, t_w, b, t_f] . \tag{2.128}$$

2.5. Távvezetéktorony optimálási modellje

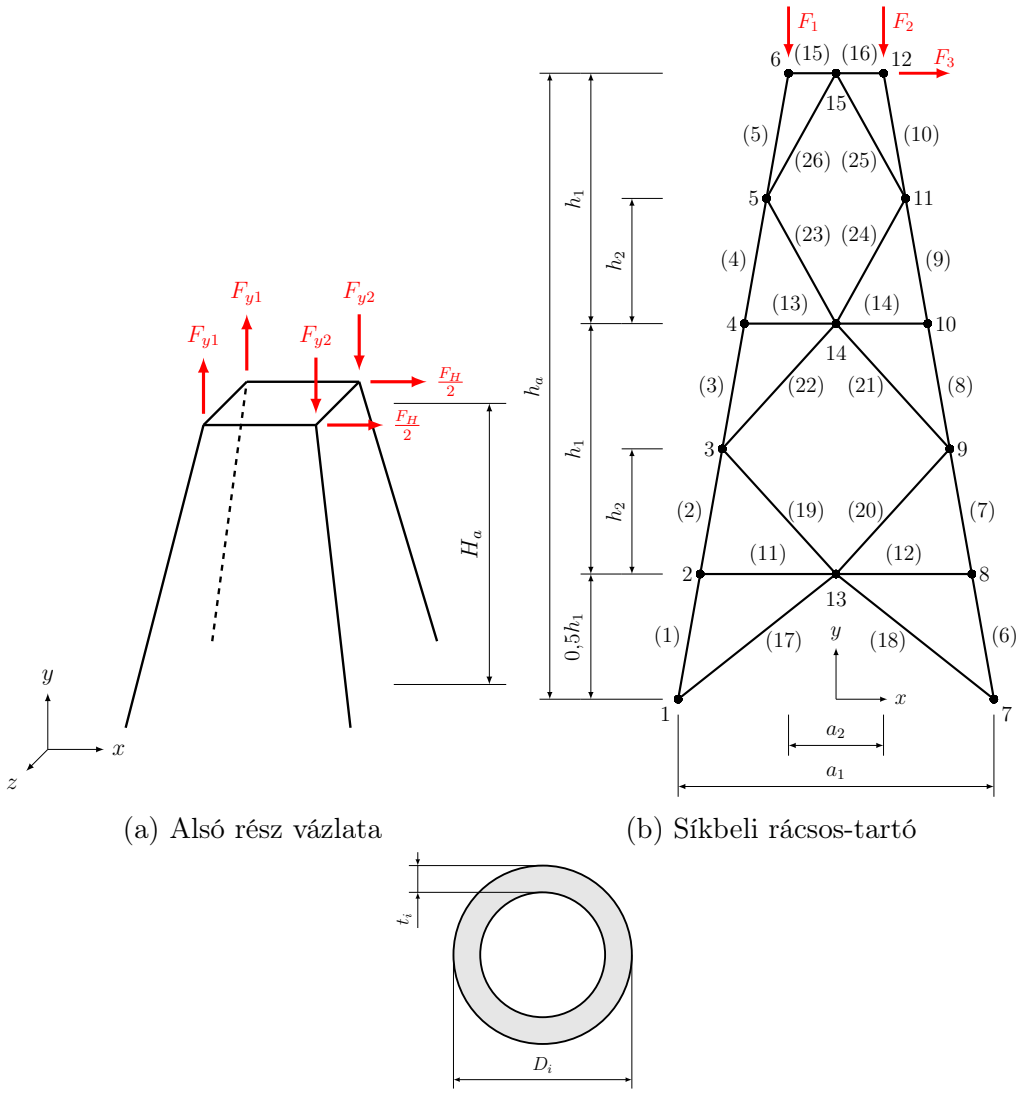
Az acél távvezetéktornyok a 2.7. ábra rendszerint két részre oszthatóak. Egy csonka gúla formájú rácsos alsó félre, és egy, a vezetékek rögzítésére szolgáló felső részre. Az utóbbiak kialakítása változatos formát vehet fel (lásd 2.7a. ábra), ezért jelen esetben csak az alsó rész (lásd 2.7b. és 2.8. ábrák) optimálásával foglalkozom.

(a) Eltérő kialakítású távvezetéktornyok



(b) Távvezetéktorony alsó része

2.7. ábra. Távvezetéktorony



(a) Alsó rész vázlata (b) Síkbeli rácsos-tartó

(c) Körcső-szelvény

2.8. ábra. Távvezeték torony alsó rész

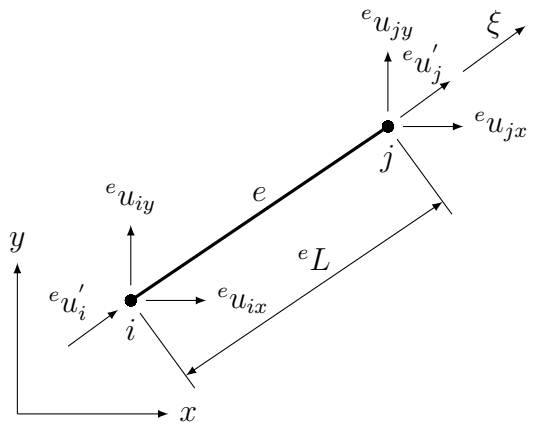
2.5.1. Végeselem-modell

A rácsos szerkezetű tartók esetében a csomópont-csatlakozásokat csuklós kapcsolattal modellezzük [118]. Csomóponti excentricitások csak a tartót alkotó elemek metszéspontjából adódnak. A csomóponti excentricitások nyomatékot generálnak, melyek a számítások során figyelmen kívül hagyhatóak, ha az excentricitás értéke $-0{,}55D \leq e \leq 0{,}25D$ határok között van CHS keresztmetszet esetén. A merev csomópontokkal történő számítás nem javasolt, még abban az esetben sem, ha hegesztett kötéssel kapcsolódnak egymáshoz [118], mert túlzott nyomatékot generálnak a rácsrudakban.

Sík és térbeli rácsos tartók modellezhetőek húzott-nyomott rúdelemekkel (2.9. ábra). Csomóponti elmozdulás csak az $i.$ és $j.$ csomópontokon átmenő ξ tengely (helyi koordináta-rendszer) mentén lehetséges. A szerkezethez kötött x-y globális koordináta-rendszerben pedig ezen elmozdulás x, y vetületei értelmezettek.

A rúdelemen belüli elmozdulást közelítsük az

$$ {}^{e}u(\xi) = \begin{bmatrix} \frac{\xi_i - \xi}{{}^{e}L} & \frac{\xi_j - \xi}{{}^{e}L} \end{bmatrix} \begin{bmatrix} {}^{e}u_i' \\ {}^{e}u_j' \end{bmatrix} = [{}^{e}N_i(\xi)\ {}^{e}N_j(\xi)] \begin{bmatrix} {}^{e}u_i' \\ {}^{e}u_j' \end{bmatrix} = {}^{e}\mathbf{N}\,{}^{e}\mathbf{u}' \tag{2.129} $$



2.9. ábra. Húzott-nyomott síkbeli rúdelem

függvénnyel, amely kinematikailag lehetséges, ahol ${}^{e}\mathbf{N}$ alakfüggvények mátrixa és ${}^{e}\mathbf{u}'$ a rúdhoz kötött lokális koordináta-rendszerben értelmezett csomóponti elmozdulások vektora [33]. A globális-koordináta rendszerben pedig a csomóponti elmozdulások vektora

$$ {}^{e}\mathbf{u} = \begin{bmatrix} {}^{e}u_{ix} & {}^{e}u_{iy} & {}^{e}u_{jx} & {}^{e}u_{jy} \end{bmatrix}^{T} \tag{2.130} $$

alakban írható fel. A két koordináta-rendszer között az átjárás a transzformációval lehetséges

$$ {}^{e}\mathbf{T} = \begin{bmatrix} {}^{e}T_{11} & {}^{e}T_{12} & 0 & 0 \\ 0 & 0 & {}^{e}T_{23} & {}^{e}T_{24} \end{bmatrix}, \tag{2.131} $$

$$ {}^{e}T_{11} = {}^{e}T_{23} = \frac{{}^{e}u_{jx} - {}^{e}u_{ix}}{{}^{e}L}, \qquad {}^{e}T_{12} = {}^{e}T_{24} = \frac{{}^{e}u_{jy} - {}^{e}u_{iy}}{{}^{e}L}, \tag{2.132} $$

$$ {}^{e}\mathbf{u}' = {}^{e}\mathbf{T}\,{}^{e}\mathbf{u}. \tag{2.133} $$

A rúdelem fajlagos nyúlása

$$ {}^{e}\varepsilon = \frac{d\,{}^{e}u(\xi)}{d\xi} = \frac{1}{{}^{e}L}\begin{bmatrix} -1 & 1 \end{bmatrix} {}^{e}\mathbf{u}', \tag{2.134} $$

továbbá a tengelyirányú normálfeszültség

$$ {}^{e}\sigma = E\,{}^{e}\varepsilon = \frac{E}{{}^{e}L}\begin{bmatrix} -1 & 1 \end{bmatrix} {}^{e}\mathbf{u}'. \tag{2.135} $$

Egy ${}^{e}A$ keresztmetszetű prizmatikus rúdelem alakváltozási energiája

$$ \begin{aligned} {}^{e}U &= \frac{1}{2}\int_{L} {}^{e}A\,{}^{e}\sigma\,{}^{e}\varepsilon\; d\xi = \frac{1}{2}{}^{e}\mathbf{u}'^{T}\int_{L}\frac{1}{{}^{e}L}\begin{bmatrix} -1 \\ 1 \end{bmatrix} {}^{e}AE\frac{1}{{}^{e}L}\begin{bmatrix} -1 & 1 \end{bmatrix}\; d\xi\,{}^{e}\mathbf{u}' = \\ &= \frac{1}{2}{}^{e}\mathbf{u}'^{T}\frac{{}^{e}AE}{{}^{e}L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} {}^{e}\mathbf{u}' = \frac{1}{2}{}^{e}\mathbf{u}'^{T}\,{}^{e}\mathbf{K}'\,{}^{e}\mathbf{u}' \end{aligned}, \tag{2.136} $$

ahol ${}^{e}\mathbf{K}'$ az elem merevségi mátrixa. A külső erők munkája pedig

$$ {}^{e}W = \int_{L} {}^{e}u(\xi)p\; d\xi = {}^{e}\mathbf{u}'^{T}\int_{L} {}^{e}N(\xi)p\; d\xi = {}^{e}\mathbf{u}'^{T}\,{}^{e}\mathbf{f}', \tag{2.137} $$

ahol ${}^e\mathbf{f}'$ a külső terhelések csomópontba redukált elemekhez tartozó terhelés vektora. Egy elem teljes potenciális energiája

$$ {}^e\Pi_p = {}^eU - {}^eW = \frac{1}{2} {}^e\mathbf{u}'^T \, {}^e\mathbf{K}' \, {}^e\mathbf{u}' - {}^e\mathbf{u}'^T \, {}^e\mathbf{f}' \tag{2.138} $$

szerint alakul, globális-koordináta rendszerben értelmezett mennyiségekkel pedig

$$ {}^e\Pi_p = \frac{1}{2} {}^e\mathbf{u}^T \, {}^e\mathbf{K} \, {}^e\mathbf{u} - {}^e\mathbf{u}^T \, {}^e\mathbf{f} \,, \tag{2.139} $$

ahol

$$ {}^e\mathbf{K} = {}^e\mathbf{T}^T \, {}^e\mathbf{K}' \, {}^e\mathbf{T} \,, \qquad {}^e\mathbf{f} = {}^e\mathbf{T}^T \, {}^e\mathbf{f}' \,. \tag{2.140} $$

Bevezetve az $\mathbf{u}$ szerkezeti csomóponti elmozdulásvektort, és az $\mathbf{f}$ összes csomóponti terhelések vektorát a teljes szerkezet potenciális energiája

$$ \Pi_p = \frac{1}{2} \mathbf{u}^T \left(\mathbf{K} \, \mathbf{u} - \mathbf{f} \right) \,, \tag{2.141} $$

ahol $\mathbf{K}$ az elemillesztés szabályait (lásd [33, 83, 104] irodalmakban) betartva a teljes szerkezet merevségi mátrixa. Egyensúlyban a Π_p potenciális energia minimális, ha az első $\delta\Pi_p$ variációja zérus [11, 83, 84]. A peremfeltételek (megfogások, stb.) alkalmazása mellett a $\delta\Pi_p = 0$ egyenlet értelmében a megoldandó algebrai egyenletrendszer

$$ \delta\Pi_p = \delta\mathbf{u}^T \frac{\partial \Pi_p}{\partial \mathbf{u}} = \delta\mathbf{u}^T \left(\mathbf{K}\mathbf{u} - \mathbf{f} \right) = 0 \qquad \longrightarrow \qquad \mathbf{K}\mathbf{u} = \mathbf{f} \,. \tag{2.142} $$

A (2.129) egyenlettől míg eljutunk a végeselem-módszer szerint kapott (2.142) egyenletig, az út több lépésből és számításból áll. Továbbá újabb számítás (utófeldolgozás), sorozat szükséges ahhoz, hogy a kapott $\mathbf{u}$ csomóponti elmozdulásvektorból a rácsos tartó méretezéséhez szükséges feszültségeket kapjuk

$$ {}^e\sigma = \frac{{}^eE}{{}^eL} \begin{bmatrix} -{}^eT_{11} & -{}^eT_{12} & {}^eT_{11} & {}^eT_{12} \end{bmatrix} {}^e\mathbf{u}^T \,. \tag{2.143} $$

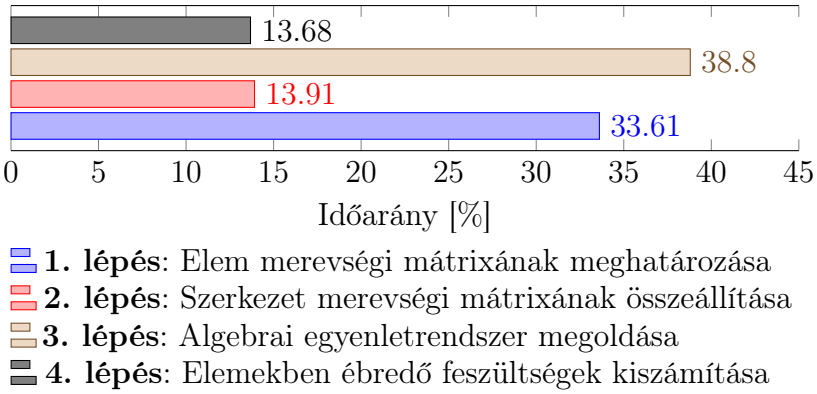
A 2.10. ábra szemlélteti ezeket a lépéseket a kiszámításukhoz szükséges időaránnyal. Az időarányok meghatározásánál a 2.8b. ábrán szereplő szerkezet megoldásához szükséges idők lettek figyelembe véve. Jelen elem- és csomópontszámmal a legnagyobb erőforrást, időt igénylő számítások: az elemek merevségi mátrixának meghatározása és az egyenletrendszer megoldása. A továbbiakban előforduló maximális csomópont és elemszám is alacsony, ezért a 2.10. ábrán vázolt arányok jól közelítik azt az esetet is. Ha a csomópontok száma nagy lenne, akkor a teljes számítási igény nagy százalékát a 3. lépés tenné ki.

Egy-egy optimáláson belül a szerkezet struktúrája nem változik, csak a keresztmetszeti területek, ezért érdemes elemenként annyi „félkész" merevségi mátrixot definiálni, ahány keresztmetszeti csoport van

$$ {}^e\hat{\mathbf{K}}_i = \begin{cases} {}^e\mathbf{K_i}/A_i & \text{ha } i. \text{ keresztmetszet csoportba tartozik az elem} \\ \mathbf{0} & \text{egyébként} \end{cases} \,. \tag{2.144} $$

A ${}^e\hat{\mathbf{K}}_i$ mátrixok felhasználásával összeállítható keresztmetszetenként a szerkezet „félkész" $\hat{\mathbf{K}}_i$ merevségi mátrixa. Ebből is annyi készül, ahány keresztmetszetcsoport van. A végeselemes feladat megoldásához szükséges $\mathbf{K}$ mátrix pedig az alábbiak szerint számítható

$$ \mathbf{K} = A_1\hat{\mathbf{K}}_1 + A_2\hat{\mathbf{K}}_2 + A_3\hat{\mathbf{K}}_3 + \ldots + A_i\hat{\mathbf{K}}_i \,. \tag{2.145} $$



2.10. ábra. Végeselem-módszer megoldásához szükséges lépések

Könnyen belátható, hogy ezzel a módszerrel elkerülhető, hogy az optimalizálás során ki kelljen számítani minden egyes egyedre az elemenkénti merevségi mátrixot. Elég csak egyszer, az elején, az iterációs ciklus megkezdése előtt. Továbbá a $\hat{\mathbf{K}}_1, \hat{\mathbf{K}}_2, \ldots, \hat{\mathbf{K}}_i$ merevségi mátrixokon már a megfogásokból származó csomóponti elmozdulások is alkalmazhatók, amivel nemcsak újabb matematikai műveletek spórolhatók meg, hanem programozástechnikailag szükséges lassú memóriamanipulációk is, mint például: új memóriaterület lefoglalás, illetve már meglévők felszabadítása. A kereskedelmi szoftverek - mint például: ADINA, ANSYS - ezt az egyszerűsítést nem támogatják.

Algoritmus 7: Végeselem-modul előfeldolgozójának pszeudokódja

Input: adatfájl neve
Output: $\hat{\mathbf{K}}_1, \hat{\mathbf{K}}_2, \hat{\mathbf{K}}_3, \ldots, \hat{\mathbf{K}}_n$ merevségi mátrixok

1 Szöveges adatfájl beolvasása
2 Rúdelemek ${}^eL$ hosszának meghatározása
3 Elemenként ${}^e\mathbf{T}$ transzformációs mátrixok meghatározása
4 Elemek ${}^e\hat{\mathbf{K}} = {}^e\mathbf{K} \,/\, {}^eA$ merevségi mátrixának meghatározása
5 Keresztmetszet csoportonként a teljes szerkezetre vonatkozóan $\hat{\mathbf{K}}_i$ merevségi mátrixok felépítése
6 Peremfeltételek alkalmazása

A végeselem-modellt felhasználó optimalizáláshoz saját programot, függvényeket fejlesztettem[1], melyek képesek egyedi szintaktikájú adatfájlból rúdelemeket tartalmazó végeselemes-modellt felépíteni, és a kapcsolódó egyenleteket megoldani. A program két részre bontható. Első része az előfeldolgozásért felel, és az optimáláshoz szükséges $\hat{\mathbf{K}}_i$ mátrixokat állítja elő (lásd 7. algoritmus). Magát a véges-elemes feladatot a (2.142) és (2.143) egyenleteket nem oldja meg. Ezek kiszámítása egyedenként a fitneszfüggvényen belül történik.

A kifejlesztett VEM program és függvények működése validálásra került. A validációhoz a 2.8b. ábrán látható szerkezet csomóponti elmozdulási és a rudak axiális irányú belső erőit határoztam meg. ADINA szoftverrel és a saját VEM programmal. A 2.8b. ábra jelöléseivel az alkalmazott terhelő erők: $F_1 = 332{,}94\,\text{kN}$, $F_2 = -437{,}46\,\text{kN}$ és $F_3 = = 312{,}143\,\text{kN}$. Az alkalmazott megfogások pedig 1 és 7 csomópontokban az x, y tengelyű csomóponti elmozdulás zérus. A szerkezetet alkotó rudak három keresztmetszetcsoportot alkotnak, amit a 2.1. táblázat foglal össze az őket jellemző ${}^eA^eE$ szorzatot is feltüntetve.

[1] A forráskód megtalálható a mellékelt adathordozón `./6_sz_melleklet/` könyvtárban

2.1. táblázat. Rácsos tartó keresztmetszet csoportjainak jellemzői

Km. csoport	Elem	${}^eA^eE$ [N]
1.	1-10	$801 \cdot 10^8$
2.	11-16	$878 \cdot 10^8$
3.	17-26	$605 \cdot 10^8$

Validáció során a végeselemes feladat megoldásának két eredménye került összehasonlításra, a csomóponti elmozdulások és a rúdelemekben ébredő belső tengelyirányú erők. A csomóponti elmozdulások összehasonlítását a 2.2. táblázat foglalja össze. Látható, hogy a megfogási csomópontokban a vártnak megfelelően a csomóponti elmozdulás zérus. A többi csomópontban pedig a két program által adott eredmény között a különbség kicsi, 10^{-4}–10^{-5} mm tartományba esik. Belső erők összehasonlítását 2.3. táblázat foglalja össze. A két rendszer által számolt értékek közötti különbség vagy $10^{-10}\,[N]$ nagyságrendbe esik, vagy zérus.

2.5.2. Optimálási modell

A villanyoszlop alsó része optimálisnak tekinthető, ha a költsége minimális, de a vele szemben támasztott műszaki követelmények maximálva vannak. A körcső-szelvényű keresztmetszetek költsége, mint sok más esetben is, két fő csoportra oszthatóak. Anyag és gyártási költségekre. Az ipari tapasztalatok és [49] által javasolt költségszámítási modell alapján az anyagköltség a tömeg függvénye, amit a keresztmetszeti jellemzők határoznak meg. A [49] szerinti közelítő számítások alapján a gyártási költségek erősen a csomópontok számától függnek, a keresztmetszeti jellemzőktől csak nagyon kis mértékben.

2.2. táblázat. Csomóponti elmozdulások összehasonlítása

Csomópont	ADINA		Saját VEM		Eltérés		
	u_x [mm]	u_y [mm]	u_x [mm]	u_y [mm]	u_x [mm]	u_y [mm]	Abszolút érték [mm]
1	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	$2{,}24 \cdot 10^0$	$3{,}93 \cdot 10^0$	$2{,}24 \cdot 10^0$	$3{,}93 \cdot 10^0$	$2{,}70 \cdot 10^{-4}$	0,00	$2{,}70 \cdot 10^{-4}$
3	$1{,}83 \cdot 10^1$	$6{,}14 \cdot 10^0$	$1{,}83 \cdot 10^1$	$6{,}14 \cdot 10^0$	$8{,}00 \cdot 10^{-4}$	$4{,}00 \cdot 10^{-5}$	$8{,}01 \cdot 10^{-4}$
4	$3{,}06 \cdot 10^1$	$6{,}61 \cdot 10^0$	$3{,}06 \cdot 10^1$	$6{,}61 \cdot 10^0$	$1{,}10 \cdot 10^{-3}$	$7{,}00 \cdot 10^{-5}$	$1{,}10 \cdot 10^{-3}$
5	$5{,}11 \cdot 10^1$	$5{,}62 \cdot 10^0$	$5{,}11 \cdot 10^1$	$5{,}62 \cdot 10^0$	$1{,}60 \cdot 10^{-3}$	$1{,}30 \cdot 10^{-4}$	$1{,}60 \cdot 10^{-3}$
6	$5{,}88 \cdot 10^1$	$5{,}01 \cdot 10^0$	$5{,}88 \cdot 10^1$	$5{,}01 \cdot 10^0$	$1{,}80 \cdot 10^{-3}$	$1{,}50 \cdot 10^{-4}$	$1{,}80 \cdot 10^{-3}$
7	0,00	0,00	0,00	0,00	0,00	0,00	0,00
8	$2{,}24 \cdot 10^0$	$-4{,}63 \cdot 10^0$	$2{,}24 \cdot 10^0$	$-4{,}63 \cdot 10^0$	$2{,}70 \cdot 10^{-4}$	0,00	$2{,}70 \cdot 10^{-4}$
9	$2{,}05 \cdot 10^1$	$-7{,}28 \cdot 10^0$	$2{,}05 \cdot 10^1$	$-7{,}28 \cdot 10^0$	$8{,}00 \cdot 10^{-4}$	$3{,}00 \cdot 10^{-5}$	$8{,}00 \cdot 10^{-4}$
10	$3{,}06 \cdot 10^1$	$-8{,}64 \cdot 10^0$	$3{,}06 \cdot 10^1$	$-8{,}64 \cdot 10^0$	$1{,}10 \cdot 10^{-3}$	$6{,}00 \cdot 10^{-5}$	$1{,}10 \cdot 10^{-3}$
11	$5{,}06 \cdot 10^1$	$-8{,}26 \cdot 10^0$	$5{,}06 \cdot 10^1$	$-8{,}26 \cdot 10^0$	$1{,}60 \cdot 10^{-3}$	$1{,}00 \cdot 10^{-4}$	$1{,}60 \cdot 10^{-3}$
12	$6{,}11 \cdot 10^1$	$-7{,}38 \cdot 10^0$	$6{,}11 \cdot 10^1$	$-7{,}38 \cdot 10^0$	$1{,}70 \cdot 10^{-3}$	$1{,}10 \cdot 10^{-4}$	$1{,}70 \cdot 10^{-3}$
13	$2{,}24 \cdot 10^0$	$1{,}43 \cdot 10^{-6}$	$2{,}24 \cdot 10^0$	$1{,}43 \cdot 10^{-6}$	$2{,}70 \cdot 10^{-4}$	$1{,}70 \cdot 10^{-10}$	$2{,}70 \cdot 10^{-4}$
14	$3{,}06 \cdot 10^1$	$-1{,}45 \cdot 10^0$	$3{,}06 \cdot 10^1$	$-1{,}45 \cdot 10^0$	$1{,}10 \cdot 10^{-3}$	$2{,}00 \cdot 10^{-5}$	$1{,}10 \cdot 10^{-3}$
15	$5{,}93 \cdot 10^1$	$-1{,}03 \cdot 10^0$	$5{,}93 \cdot 10^1$	$-1{,}03 \cdot 10^0$	$1{,}70 \cdot 10^{-3}$	$1{,}00 \cdot 10^{-5}$	$1{,}70 \cdot 10^{-3}$
				Átlag	$9{,}41 \cdot 10^{-4}$	$4{,}80 \cdot 10^{-5}$	$9{,}42 \cdot 10^{-4}$

2.3. táblázat. Rúdelemek tengely irányú belső erőinek összehasonlítása

Elem	Rúderő [N]		Eltérés [N]
	ADINA	Saját VEM	
0	$6{,}60 \cdot 10^5$	$6{,}60 \cdot 10^5$	0,00
1	$6{,}60 \cdot 10^5$	$6{,}60 \cdot 10^5$	0,00
2	$5{,}34 \cdot 10^5$	$5{,}34 \cdot 10^5$	0,00
3	$5{,}34 \cdot 10^5$	$5{,}34 \cdot 10^5$	0,00
4	$3{,}38 \cdot 10^5$	$3{,}38 \cdot 10^5$	0,00
5	$-7{,}66 \cdot 10^5$	$-7{,}66 \cdot 10^5$	0,00
6	$-7{,}66 \cdot 10^5$	$-7{,}66 \cdot 10^5$	0,00
7	$-6{,}40 \cdot 10^5$	$-6{,}40 \cdot 10^5$	0,00
8	$-6{,}40 \cdot 10^5$	$-6{,}40 \cdot 10^5$	0,00
9	$-4{,}44 \cdot 10^5$	$-4{,}44 \cdot 10^5$	0,00
10	$-4{,}39 \cdot 10^{-2}$	$-4{,}39 \cdot 10^{-2}$	0,00
11	$5{,}09 \cdot 10^{-2}$	$5{,}09 \cdot 10^{-2}$	0,00
12	0,00	$-4{,}65 \cdot 10^{-10}$	$4{,}65 \cdot 10^{-10}$
13	0,00	$9{,}31 \cdot 10^{-10}$	$9{,}31 \cdot 10^{-10}$
14	$5{,}87 \cdot 10^4$	$5{,}87 \cdot 10^4$	0,00
15	$2{,}35 \cdot 10^5$	$2{,}35 \cdot 10^5$	0,00
16	$4{,}52 \cdot 10^4$	$4{,}52 \cdot 10^4$	0,00
17	$-4{,}52 \cdot 10^4$	$-4{,}52 \cdot 10^4$	0,00
18	$-6{,}79 \cdot 10^4$	$-6{,}79 \cdot 10^4$	0,00
19	$6{,}79 \cdot 10^4$	$6{,}79 \cdot 10^4$	0,00
20	$-8{,}43 \cdot 10^4$	$-8{,}43 \cdot 10^4$	0,00
21	$8{,}43 \cdot 10^4$	$8{,}43 \cdot 10^4$	0,00
22	$-1{,}24 \cdot 10^5$	$-1{,}24 \cdot 10^5$	0,00
23	$1{,}24 \cdot 10^5$	$1{,}24 \cdot 10^5$	0,00
24	$-1{,}19 \cdot 10^5$	$-1{,}19 \cdot 10^5$	0,00
25	$1{,}19 \cdot 10^5$	$1{,}19 \cdot 10^5$	0,00
		Átlag	$5{,}37 \cdot 10^{-11}$

Legegyszerűbb általános alakja a szerkezet költségének

$$K = K_m + K_w = k_m \sum_{i=1}^{n_e} m_i + k_w \sum_j T_j \,, \tag{2.146}$$

ahol K_m és K_w az anyag- és gyártási költség, k_m és k_w az anyag- és a gyártási költségtényezők, a T_j pedig a gyártási idők. A 2.11. ábra jól szemlélteti a 2.8b. szerinti síkbeli rácsos tartó teljes költségének az anyag- és gyártási költségek közötti megoszlását $k_m = 1 \frac{\$}{\text{kg}}$ és $k_w = 1 \frac{\$}{\text{min}}$ költségtényezőkkel. Gyártási költségnél figyelembe vett költségekhez tartózó megmunkálási idők:

– vágási idő rúdvégenként [112]

$$T_{C,i} = \frac{2{,}5\pi D_i}{0{,}3\,(350 - 2t_i)\sin\phi_i} \,, \tag{2.147}$$

ahol D_i a CHS szelvény külső átmérője mm-ben, t_i a CHS szelvény falvastagsága mm-ben, és ϕ_i a vizsgált rúdelem csatlakozó rúddal bezárt hegyesszög,

– összeállítási, összefűzési és hegesztési idő csatlakozásonként [49]

$$T_{W,i} = C_a \Theta_i \sqrt{\kappa_i \rho_i V_i} + 1{,}3 C_w a_{w,i}^n L_{w,i}\,, \tag{2.148}$$

ahol Θ_i a bonyolultsági tényező, κ_i összehegesztendő elemek száma, ρ_i az anyag sűrűsége, V_i az összehegesztendő elemek térfogata, C_w ahegesztési technológiára jellemző konstans, n hegesztési technológiára jellemző kitevő, $a_{w,i}$ a varrat jellemző mérete, és $L_{w,i}$ a varrat hossza,

– festési idő rúdelemenként [49]

$$T_{P,i} = \Theta_{dp} \left(a_{gc} + a_{tc}\right) A_s\,, \tag{2.149}$$

ahol $\Theta_{dp} = 2$ a bonyolultsági tényező, $a_{gc} = 3{\cdot}10^{-6}\,\mathrm{min/mm^2}$, $a_{tc} = 4{,}15{\cdot}10^{-6}\,\mathrm{min/mm^2}$ technológiára jellemző konstansok, A_s a festendő felület $\mathrm{mm^2}$-ben.

A költségek összehasonlításának alapjául 2.8b. ábrán vázolt szerkezet szolgált változó számú deltoid alakú rácsosztással. Az alkalmazott CHS keresztmetszetek méreteit 2.4. táblázat tartalmazza, felhasználva a 2.1. táblázat szerinti keresztmetszet csoportba sorolást.

2.4. táblázat. Költség számításhoz alkalmazott CHS szelvények méretei

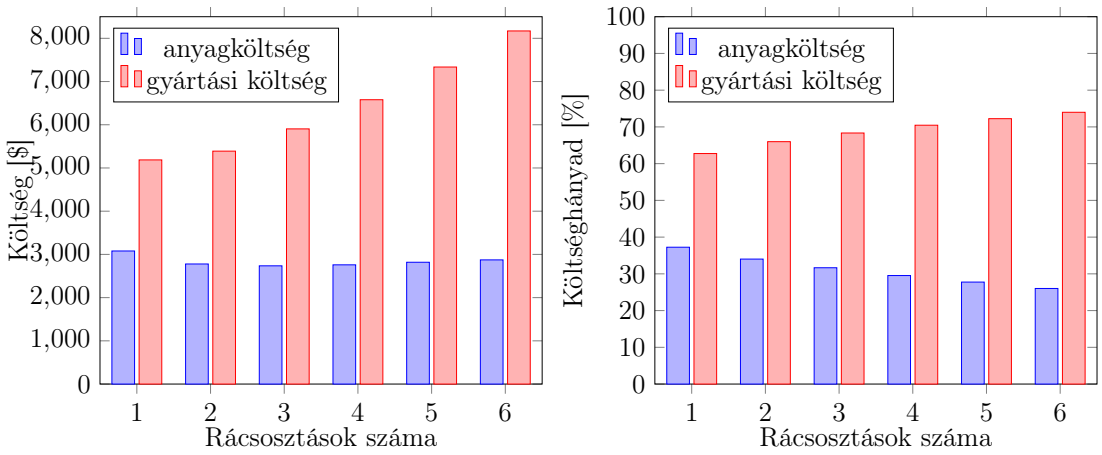
Rácsosztások száma	Keresztmetszet méretei [mm]					
	D_1	t_1	D_2	t_2	D_3	t_3
1	277	6	43	11	200	4
2	194	8	43	11	142	4
3	158	9	43	11	112	4
4	139	10	43	11	96	4
5	129	11	43	11	86	4
6	122	12	43	11	81	4

A 2.11. ábrán látható, hogy általában a nagyobb költséghányadot a gyártási költség adja. Ez a hányad rács osztások, vagyis csomópontok számának növekedésével folyamatosan nő, mind dollárban, mind százalékban kifejezve.

Az optimális szerkezet vizsgálatához elegendő lehet a tömegét optimalizálni, ha figyelembe vesszük, azt a közelítést, hogy ez az a mennyiség, amit a körcső keresztmetszetek jelentősen befolyásolnak. Továbbá a közelítő számításban elhanyagoljuk a gyártási költség keresztmetszet függőségét. Ekkor az optimális tömeget felhasználva a helyi viszonyoknak megfelelő $\frac{k_w}{k_m}$ arány segítségével becsülhető egy optimális rácsosztás szám is.

$$\min\left(\sum_{i=1}^{n_e} m_i\right) = \min\left(\rho \sum_{i=1}^{n_e} A_i L_i\right) \tag{2.150}$$

A szerkezetnek szilárdsági és stabilitási követelményeknek kell megfelelnie. Jelen esetben három kritérium került figyelembevételre. A húzott rudak esetén a húzó feszültséggel



(a) Költségek eloszlása dollárban kifejezve (b) Költségek százalékos eloszlása

2.11. ábra. Sík modell anyag- és gyártási költsége a rácsosztások függvényében

szembeni ellenállás, a nyomott rudaknál pedig a kihajlás, és végül a helyi horpadás. Ezen jellemzőket a keresztmetszeti kihasználtsági tényezővel jól lehet jellemezni.

A húzott és nyomott rudak húzással és nyomással szembeni ellenállását lehet keresztmetszet kihasználtsági tényezővel jellemezni, abban az esetben, ha a terhelésből származó feszültséget előjelesen értelmezzük. A negatív feszültség nyomást jelent, míg a pozitív húzást

$$g_{Ii} = \begin{cases} \dfrac{\gamma_{M0}|^e\sigma|}{\chi f_y} \leq 1 & {}^e\sigma < 0 \\ \dfrac{\gamma_{M0}|^e\sigma|}{f_y} \leq 1 & {}^e\sigma \geq 0 \end{cases}, \tag{2.151}$$

ahol a χ kihajlási tényező meghatározásához a [31] szabvány ajánlása került alkalmazásra

$$\chi = \frac{1}{\phi + \sqrt{\phi^2 + \bar{\lambda}^2}}, \text{ ahol} \qquad \phi = 0{,}5\left(1 + \alpha_a\left(\bar{\lambda} - 0{,}2\right) + \bar{\lambda}^2\right). \tag{2.152}$$

Az $\alpha_a = 0{,}21$ kezdeti alakpontatlansági tényező, és $\bar{\lambda}$ redukált karcsúsági tényező is [31] szerinti érték

$$\bar{\lambda} = \pi k L \sqrt{\frac{A}{I_x}} \sqrt{\frac{f_y}{E}}. \tag{2.153}$$

A k kihajlási hossztényező jelen esetben a közbenső rudakra $k = 1$, azon rudakra pedig, ahol a megfogások vannak alkalmazva $k = 0{,}7$. Nem szükséges a (2.152) egyenlettel számolni, ha fennáll a $\bar{\lambda} \leq 0{,}2$ egyenlőtlenség, mert ebben az esetben alkalmazható a $\chi = 1$ összefüggés.

A körcső-szelvények esetén helyi lemezhorpadásra [31] ajánlását felhasználva a kihasználtsági tényező

$$g_{IIi} = \frac{D f_y}{21150 t} \leq 1. \tag{2.154}$$

A (2.154) egyenlet csak abban az esetben igaz, ha a folyáshatár mértékegysége MPa.

A g_{Ii} értéket minden egyes rúdra ki kell számolni, míg a g_{IIi}-t elég csak keresztmetszet csoportonként egyszer. Könnyen belátható, hogy a szerkezet abban az esetben lesz optimális, ha a g_{Ii} a veszélyes keresztmetszet(ek)ben 1 és a g_{IIi} keresztmetszetcsoportonként úgyszintén 1.

Előzőek alapján az optimálandó fitneszfüggvény

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{n_e} m_i + \sum_{i=1}^{n_e} p_{Ii}(g_{Ii}(\mathbf{x})) + \sum_{i=1}^{3} p_{IIi}(g_{IIi}(\mathbf{x}))\,, \tag{2.155}$$

és az ismeretleneket (tervezési változókat) tömörítő vektor (egyed)

$$\mathbf{x} = [D_1, D_2, D_3, t_1, t_2, t_3]\ , \tag{2.156}$$

ahol p_{Ii} és p_{IIi} büntető-függvények:

$$p_{Ii}(\mathbf{x}) = \begin{cases} 0 & g_{Ii} \le 1 \\ 10^6 g_{Ii} & g_{Ii} > 1 \end{cases}\ , \text{ és} \qquad p_{IIi}(\mathbf{x}) = \begin{cases} 0 & g_{IIi} \le 1 \\ 10^6 g_{IIi} & g_{IIi} > 1 \end{cases}\ . \tag{2.157}$$

Az optimumban a várható eredmény 10^3 - 10^4 nagyságrendű intervallummal becsülhető. A (2.157) összefüggésben szereplő 10^6 büntetőparaméter elég messze viszi a nem lehetséges megoldásokat a lehetségesektől.

2.5.3. Számszerűsített példa

Az optimálási probléma számszerűsítéséhez egy 45 m magas, közbenső torony választottam ki. A szerkezet két részre bontható, egy $H_f = 21\,\text{m}$ magas felső-, és egy $H_a = 24\,\text{m}$ magas alsó részre.

A terhelések a [77] irodalom alapján kerültek felvételre. A mértékadó terhelés félvezetékhúzás. [77] szerinti számítások részletezése nélkül egy 400 m toronytávolságú, 12 áramvezetékes, 40 kN becsült önsúlyú felső toronyrészről alsó toronyrészre átadó terhelések: 1,1 biztonsági tényezővel számított függőleges erő $F_V = 209{,}03\,\text{kN}$, félvezetékhúzásból adódó vízszintes erő $F_H = 312{,}14\,\text{kN}$ és hajlító nyomaték $M_h = 2850{,}5\,\text{kNm}$.

A 2.8a. ábra szerinti négyzetes alapterületű, csonka gúla alakú alsó toronyrész csúcspontjaiba redukált erőrendszer

$$F_{y1} = \frac{M_h}{2a_2} - \frac{F_V}{4} = \frac{2850{,}5\,\text{kNm}}{2 \cdot 3{,}7\,\text{m}} - \frac{209{,}03\,\text{kN}}{4} = 332{,}94\,\text{kN}\,, \tag{2.158}$$

$$F_{y2} = \frac{M_h}{2a_2} + \frac{F_V}{4} = \frac{2850{,}5\,\text{kNm}}{2 \cdot 3{,}7\,\text{m}} + \frac{209{,}03\,\text{kN}}{4} = 437{,}46\,\text{kN}\,, \tag{2.159}$$

ahol $a_2 = 3{,}7\,\text{m}$ a felső toronyszélesség. A számításokat elegendő csak egy, a terhelés szempontjából lényeges ferde síkon végezni (2.8b. ábra). Egy $\beta_o = 80°$ oldalferdeségű gúla esetén a 2.8b. ábra jelöléseivel a vizsgált ferde síkra ható erők

$$F_1 = \frac{F_{y1}}{\sin\beta_o} = \frac{332{,}94\,\text{kN}}{\sin 80°} = 338{,}08\,\text{kN}\,, \qquad F_2 = \frac{F_{y2}}{\sin\beta_o} = \frac{437{,}46\,\text{kN}}{\sin 80°} = 444{,}21\,\text{kN}\,, \tag{2.160}$$

$$F_3 = \frac{F_H}{2} = \frac{312{,}14\,\text{kN}}{2} = 156{,}07\,\text{kN}\,. \tag{2.161}$$

A 2.8b. ábra egy $n_{racs} = 2$ rácsosztású példát szemléltet. A további vizsgálatok során több rácsosztású feladat is megoldásra kerül. A rácsosztások csökkentése illetve növelése a

$$h_1 = \frac{h_a}{2{,}5 n_{racs}} \tag{2.162}$$

összefüggés alapján történik. Az így kapott n_{racs} darab egyenlő rész deltoid rácsozást kap vízszintes összekötő rudakkal. Az utolsó, alsó $0{,}5h_1$ magasságú rész pedig egy, az egyik oldalán nyitott, háromszög rácsozást kap. Ez a megfontolás lehetővé teszi a több csak rácsozás számában eltérő topológia egyszerű összehasonlítását. A több vagy kevesebb rácsosztású topológiák esetén a 2.8b. ábra jelölésétől a rudak és csomópontok sorszámozása eltér, de a továbbiakban az azonos funkciójú rudak azonosítása a 2.8b. ábra jelöléseivel fog történni.

A sík rácsos tartót alkotó rudakat három csoportba soroltam. Első csoportot az oldalsó oszlopelemek alkotják (2.8b. ábra jelöléseivel 1 - 10 rudak). Másodikat a vízszintes rácsozás elemei (2.8b. ábra jelöléseivel 17 - 22 rudak). Végül a harmadik csoportba deltoidot és háromszöget alkotó rácselemek (2.8b. ábra jelöléseivel 23 - 36 rudak) tartoznak. A minden keresztmetszetcsoportban a körcső-szelvény került alkalmazásra, mely jellemző méreteit a 2.8c. ábra szemlélteti, úgymint a D_1, D_2, D_3 külső átmérők, és a t_1, t_2, t_3 falvastagságok az alkalmazott csoportok szerint.

3. fejezet

Evolúciós algoritmusok szimulációs összehasonlítása

Az evolúciós algoritmusok jól használható eljárások sokdimenziós, nemlineáris optimalizációs feladatok megoldására. Ennek ellenére felmerül a kérdés, hogy mennyire jól használhatóak, és az egyes eljárások teljesítményében milyen különbség mutatkozik.

Az algoritmusokat bemutató publikációk [13, 18, 35, 56, 58, 59, 66, 76, 93, 94, 96, 99, 101, 102, 103, 107, 119, 120, 121, 123, 124] rendszerint tartalmaznak összehasonlítást más, már meglévő algoritmusokkal, de ezek összességében nem egységesek. Külön-külön értelmezhető információt adnak egy-egy algoritmusról, de más, a publikációban nem szereplő algoritmus teszteredményeivel nehezen, vagy egyáltalán nem összehasonlíthatóak. Az összehasonlíthatóságot nehezítő körülmények sokaságából kettőt emelnék ki. Az első, hogy a szimulációkhoz használt tesztfüggvények és a keresési tér dimenziója nem azonos. A másik pedig, hogy az iteráció befejezését jelentő feltétel általában egy maximális iterációszám.

A maximális iterációszám, mint kilépési feltétel nem biztosít egyenlő feltételt, mert vannak algoritmusok, amik egy iterációs lépésen belül egyedenként nem csak egyszer határozzák meg a célfüggvény értékét az adott pontban (például: ABC, FF, IWO). Az ilyen szimulációk téves következtetésre adhatnak lehetőséget. Könnyen belátható, hogy az ilyen többszöri függvényérték kiszámolással dolgozó algoritmusok könnyen jobbak lehetnek a társaiknál. Holott csak annyiról van szó, hogy több pontot vizsgáltak meg, és nagyobb esélyük volt kiválasztani az optimumot.

Az összehasonlítás során a bevezetésben korábban felsorolt 11 algoritmust a [67] irodalomban található benchmark függvényekkel vizsgáltam. A gyűjtemény 30 tesztfüggvényt tartalmaz, melyek az alábbiak szerint csoportosíthatóak (a függvények jelöléséhez a [67] irodalom jelöléseit alkalmazom, úgy mint $F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_{30}(\mathbf{x})$ és ezek alapfüggvényei $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{14}(\mathbf{x})$):

- *konvex függvények* ($F_1(\mathbf{x})$ – $F_3(\mathbf{x})$): egyszerű konvex függvények, melyek egy globális minimumot tartalmaznak. Hagyományos numerikus deriváltakon alapuló módszerekkel is könnyen optimalizálhatóak. Általános alakjuk

$$F_n(\mathbf{x}) = f_n\left(\mathbf{M}_n\left(\mathbf{x} - \mathbf{o}_n\right)\right) + F_{n,0}\,, \tag{3.1}$$

  ahol $\mathbf{M}_n$ ortogonális forgató tenzor, $\mathbf{o}_n$ az optimum eltolásának mértéke az origóhoz képest, $F_{n,0}$ választott konstans.

- *zajos függvények* ($F_4(\mathbf{x})$ – $F_{16}(\mathbf{x})$): szakaszonként konvex, sok lokális minimumot tartalmazó zajos függvények. Hagyományos numerikus deriváltakon alapuló mód-

szerekkel nehezen, vagy egyáltalán nem optimálhatóak. Néhány esetben a lokális minimum értéke nagyon hasonló a globális minimum értékhez. Általános alakjuk

$$F_n(\mathbf{x}) = f_n\left(\mathbf{M}_n\left(\mathbf{x} - \mathbf{o}_n\right)\right) + F_{n,0}\,, \tag{3.2}$$

ahol $\mathbf{M}$ ortogonális forgató tenzor, $\mathbf{o}_n$ az optimum eltolásának mértéke az origóhoz képest, $F_{n,0}$ választott konstans.

– *hibrid függvények* ($F_{17}(\mathbf{x})$ – $F_{22}(\mathbf{x})$): a valós optimalizálási feladatokban – többek között a mérnöki gyakorlatban is – a különböző részösszetevők eltérő súllyal, tulajdonsággal vesznek részt. A hibrid függvények ezt a tulajdonságot szimulálják. Általános alakjuk

$$F_n(\mathbf{x}) = \sum_{i=1}^{N} f_i\left(\mathbf{M}_i\left(\mathbf{z}_i - \mathbf{o}_n\right)\right) + F_{n,0}\,, \tag{3.3}$$

ahol $\mathbf{M}_i$ ortogonális forgató tenzor, $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_i, \ldots, \mathbf{z}_N$ vektorok az eredeti $\mathbf{x}$ vektor elemeinek véletlen permutációjából képezett vektorok, elemszámuk kisebb vagy egyenlő az eredeti vektor elemszámával, és $F_{n,0}$ választott konstans. Ezek a függvények $n_D = 2$ esetre nincsenek értelmezve.

– *kompozit függvények* ($F_{23}(\mathbf{x})$ – $F_{30}(\mathbf{x})$): több függvény súlyozott összegeként áll össze, általános alakjuk

$$F_n(\mathbf{x}) = \sum_{i=1}^{N} \left(\omega_i\left(\lambda_i f_i(\mathbf{x} - \mathbf{o}_n) + f_{i,0}\right) + F_{n,0}\right)\,, \tag{3.4}$$

ahol $f_i(\mathbf{x})$ az i. alapfüggvény, $f_{i,0}$ az i. alapfüggvény minimum értéke, λ_i a skálázási paraméter, $F_{n,0}$ választott konstans, ω_i súlytényező, melyre igaz

$$\sum_{i=1}^{N} \omega_i = 1\,. \tag{3.5}$$

A $F_{29}(\mathbf{x})$ és $F_{30}(\mathbf{x})$ függvények $n_D = 2$ esetben nincsenek értelmezve.

[67] minden tesztfüggvény számára a szükséges konstansokat, eltolásokat is rögzíti.

3.1. táblázat. Alkalmazott n_p populációméretek

	n_D				
	2	10	30	50	100
n_p	20	40	60	100	100

Minden tesztfüggvény minden optimáló algoritmussal öt különböző problémanagyság (változószám) mellett került megvizsgálásra, úgymint $n_D = 2$, $n_D = 10$, $n_D = 30$, $n_D = 50$ és $n_D = 100$. A „no-free lunch” [25] elmélet értelmében és figyelembe véve, hogy az evolúciós algoritmusok által adott eredmények nem determinisztikusak – mivel erősen függenek a véletlentől – egy-egy szimuláció függvényenként és probléma nagyságonként nem elegendő. Minden szimulációt ötvenszer ismételtem. A populációk nagysága algoritmusonként azonos, de a problémanagyságonként eltérő volt. Az alkalmazott populációméreteket

a 3.1. táblázat foglalja össze, ahol a szükséges egyedek számát tapasztalati úton becsültem. Az algoritmusok belső működését (sikerességüket) befolyásoló paraméterek értéke a már korábban hivatkozott irodalmi ajánlásokat követi, az alkalmazott értékeket a 3.2. táblázat foglalja össze. Az iterációs ciklusból a kilépési feltétel maximális alkalommal a fitneszfüggvény függvényértékének meghatározása volt

$$n_{FE} = 10^5 n_D \,. \tag{3.6}$$

Ezzel is biztosítva az azonos szimulációs körülményeket, mert az iterációs ciklus felépítésétől függően egyedenként akár többször is meg kell határozni a fitnesz értékét.

3.2. táblázat. Alkalmazott n_p populációméretek

Algoritmus		Paraméterek
ABC	mesterséges méhkolónia optimalizáció	dolgozó méhek száma: $n_{rb} = n_p$ bámészkodó méhek száma: $n_{ob} = n_p$ egy méh (egyed) élethossza: $0{,}6 n_D n_p$
BBO	bio-geográfiai optimalizáció	véletlen mozgás valószínűsége: $p_p = 0{,}1$
DE	differenciális evolúció	mutációs stratégia: DE/rand/1 keresztezési tényező: $C_R = 0{,}5$ skálázási tényező: $F_S = 0{,}9$
FA	szentjánosbogár algoritmus	fényelnyelődési tényező: $\gamma_l = 0{,}01$ vonzódási konstans: $\beta_l = 1$ véletlen mozgás konstansa: $\alpha_l = 0{,}97$ konstans: $\alpha_{damp} = 0{,}05$ kitevő: $m = 2$
FPA	virágbeporzási algoritmus	valószínűségi változó: $p_p = 0{,}5$ paraméter: $\lambda = 1{,}5$
HS	harmónia keresés	konszolidációs ráta: $p_p = 0{,}9$ hangjegy hangolási ráta: $p_{pa} = 0{,}1$ skálázási konstans: $\alpha_{damp,i} = 0{,}02(x_{U,i} - x_{L,i})$
IWO	invazív gyom optimalizáció	σ szórás kezdő értéke: $\sigma_{init,i} = 0{,}5(x_{U,i} - x_{L,i})$ σ szórás vég értéke: $\sigma_{final,i} = 0{,}001(x_{U,i} - xL, i)$ reprodukciós tényező: $n_{SR} = 0{,}25 n_p$
PSO	részecskeraj optimalizáció	konstansok: $\phi_1 = 2{,}05$, $\phi_2 = 2{,}05$
SaDE	önadaptív differenciális evolúció	mutáció tanulási ciklusszáma: $n_{lp} = 50$ C_R keresztezési tényező tanulási ciklusszáma: $n_{lCR} = 5$
SaNSDE	önadaptív differenciális evolúció szomszédsági kereséssel	mutáció tanulási ciklusszáma: $n_{lp} = 50$ C_R keresztezési tényező tanulási ciklusszáma: $n_{lCR} = 5$
StdBA	méh algoritmus	dolgozó méhek száma: $n_{rb} = 0{,}4 n_p$ bámészkodó méhek száma: $n_{ob} = 0{,}5 n_p$ felderítő méhek száma: $n_{sb} = 0{,}1 n_p$ megvizsgált források száma: $n_{rs} = 2n_{os}$, $n_{os} = 0{,}5 n_p$ keresési kör sugara: $r_{1,i} = r_{2,i} = 0{,}1(x_{U,i} - x_{L,i})$

A szimulációkat, és azok eredményének a feldolgozását MatLAB környezetben írt programok segítségével végeztem[2]. Az eredmények összehasonlításához két mutatószámot definiáltam:

[2] A forráskód megtalálható a mellékelt adathodozón `./3_sz_melleklet/` könyvtárban

– *átlagos hibaérték*: függvényenként és probléma nagyságonként az 50 szimulációs eredmény hibaátlaga az ismert optimumhoz képest j. függvény esetén

$$\overline{\varepsilon}_{avg,j} = \frac{1}{50}\sum_{i}^{50} \mathcal{F}_j(\mathbf{x}_i) - \mathcal{F}_{j,min}\,, \tag{3.7}$$

ahol $\mathcal{F}_j(\mathbf{x}_i)$ a j. tesztfüggvény megtalált optimuma az i. szimulációban, $\mathcal{F}_{j,min}$ a j. tesztfüggvény ismert optimuma;

– *legjobb hibaérték*: függvényenként és probléma nagyságonként a legkisebb hiba érték az 50 szimulációs eredményből ismert optimumhoz képest

$$\overline{\varepsilon}_{min} = \min\left(\mathcal{F}_j(\mathbf{x}_1); \mathcal{F}_j(\mathbf{x}_2); \ldots; \mathcal{F}_j(\mathbf{x}_i); \ldots; \mathcal{F}_j(\mathbf{x}_{50})\right) - \mathcal{F}_{j,min}\,. \tag{3.8}$$

A szimulációk átlagos hibaértékének konvergenciáját a függvényérték meghatározások számának függvényében CD melléklet[3] tartalmazza. A CD melléklet[4] tartalmazza a hibaértékek eloszlását gyertyadiagramon ábrázolva, algoritmusonként lebontva a különböző tesztfüggvényekre és problémanagyságokra. A logaritmikus ábrázolás korlátai miatt, ha a medián nulla, akkor a gyertyadiagram hiányzik. Ilyenkor csak a nem nulla hibaértékek kerültek ábrázolásra.

A 3.1. ábrán látható, hogy problémanagyságonként a teljes tesztkészletre nézve az esetek hány százalékában volt az átlagos hibaérték kicsi ($\overline{\varepsilon}_{avg} \leq 10^0$), közepes ($10^0 < \overline{\varepsilon}_{avg} \leq 10^1$), illetve nagy ($10^1 < \overline{\varepsilon}_{avg}$). Ezen három csoportot felhasználva megfigyelhető, hogy átlagosan egy adott probléma nagyságnál megbízható-e az algoritmus. Továbbá a 3.1. ábráról az is leolvasható, hogy az esetek hány százalékában talált az algoritmus a függvényenkénti 50 futtatásból legalább egyszer optimumot ($\overline{\varepsilon}_{min} \leq 10^0$), közel járt hozzá ($10^0 < \overline{\varepsilon}_{min} \leq 10^1$), vagy egyáltalán nem sikerült ($10^1 < \overline{\varepsilon}_{min}$).

Mind az iterációs görbékből, mind a hibaértékek eloszlásából az tűnik ki, hogy kétváltozós ($n_D = 2$) feladatokat nagy pontossággal képesek optimálni, viszont a változószám növekedéssel megbízhatóságuk kezd csökkenni. Egyre lassabb a konvergencia sebessége, és/vagy egyre nagyobb valószínűséggel találnak lokális minimumot a globális helyett. Az $n_D = 100$ esetben olyan is előfordul, hogy egyáltalán nem történik optimálás.

Összességében elmondható, hogy kis méretű problémák esetén $n_D = 2 - 10$ a vizsgált algoritmusok, az alkalmazott populációmérettel és a belső paraméter beállításokkal elfogadható mértékben használhatóak. Ez nem jelenti azt, hogy minden további feladat esetében hasonló eredményt adnak. A megbízhatóságuk tovább javítható a feladatspecifikus finomhangolással, úgymint a populáció méretének, és/vagy belső paraméterek megváltoztatása, és/vagy n_{FE} függvényérték meghatározások számának növelése.

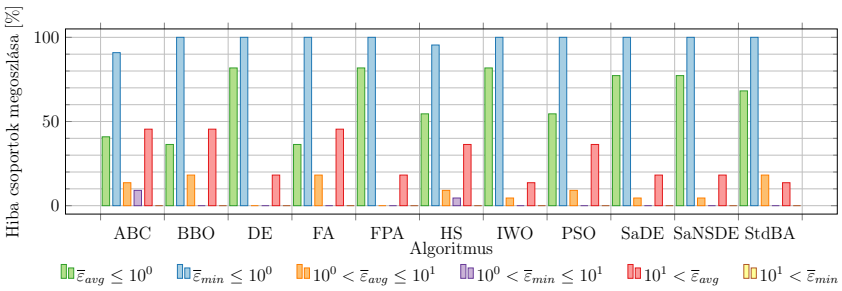
Az $\overline{\varepsilon}_{avg}$ átlagos hibaértékek alapján rangsorolásra kerültek problémanagyságonként, és azon belül is függvényenként az algoritmusok az alábbi szabályok szerint:

– 0 a rangja az algoritmusnak, ha nincs értelmezve az adott függvény, és nem számít bele ez a rang az átlagos rangba,

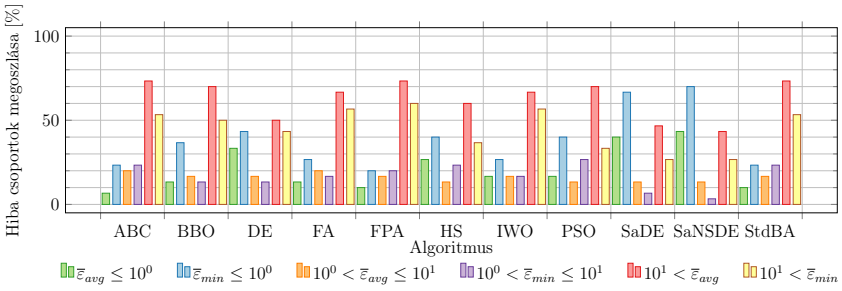
– a legkisebb $\overline{\varepsilon}_{avg}$ értékhez az 1-es rang tartozik, és a további rangok növekvő sorban jönnek $\overline{\varepsilon}_{avg}$ függvényében,

– két megegyező átlagos hibaérték esetén a rang is azonos.

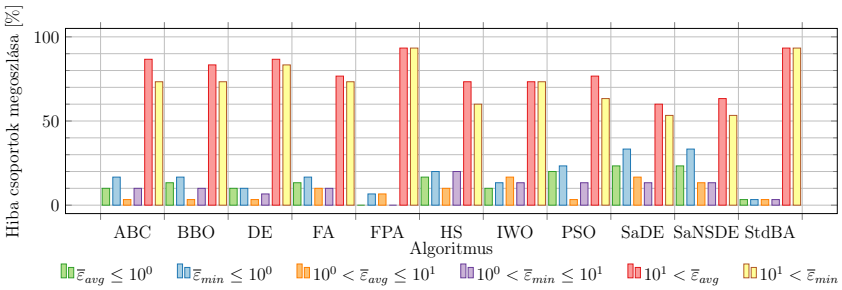
[3] Megtalálható a mellékelt adathordozón `./1_sz_melleklet/` könyvtárban
[4] Megtalálható a mellékelt adathordozón `./2_sz_melleklet/` könyvtárban



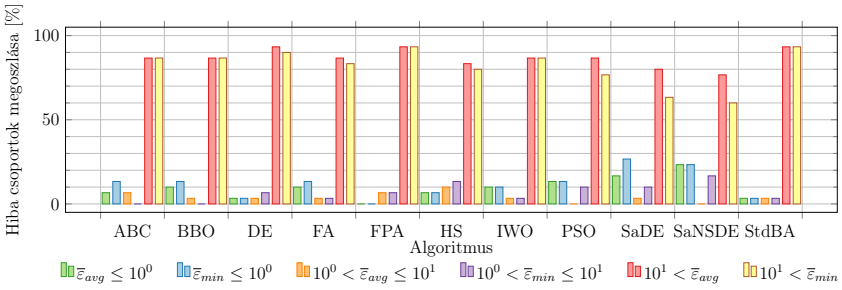
(a) $n_D = 2$



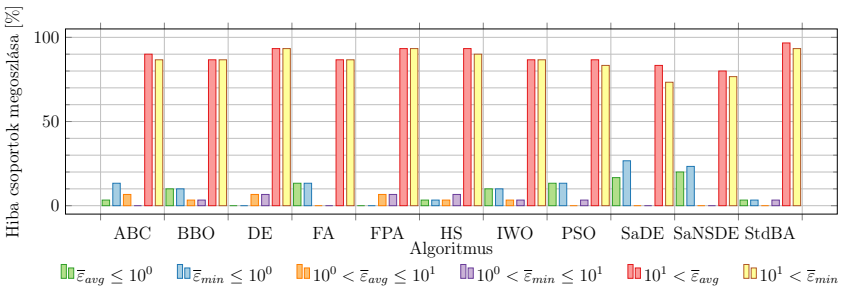
(b) $n_D = 10$



(c) $n_D = 30$

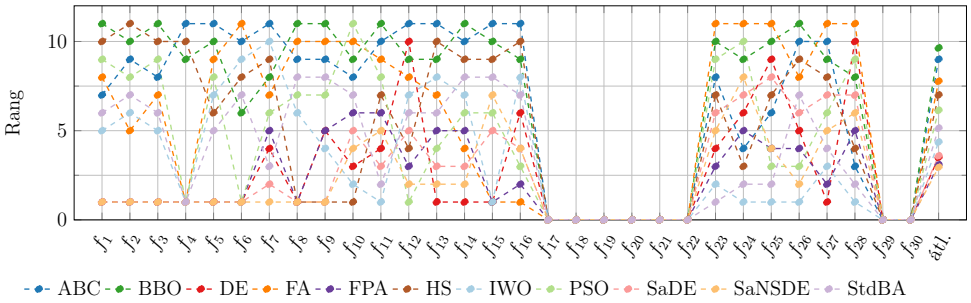


(d) $n_D = 50$

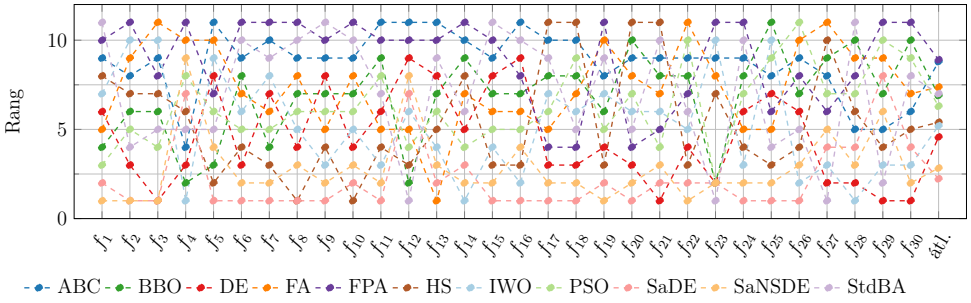


(e) $n_D = 100$

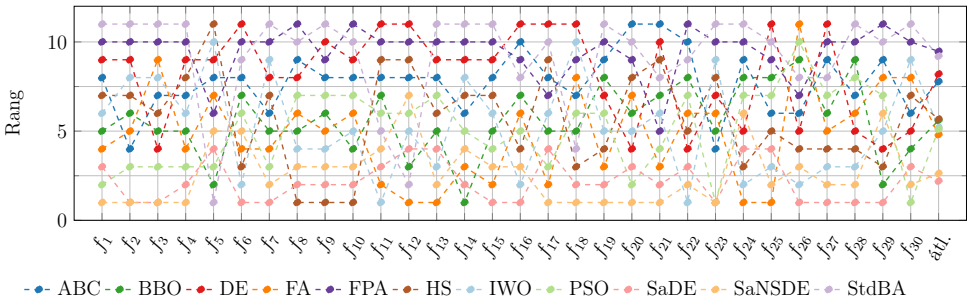
3.1. ábra. Hibaértékek százalékos megoszlása különböző n_D értékek esetén



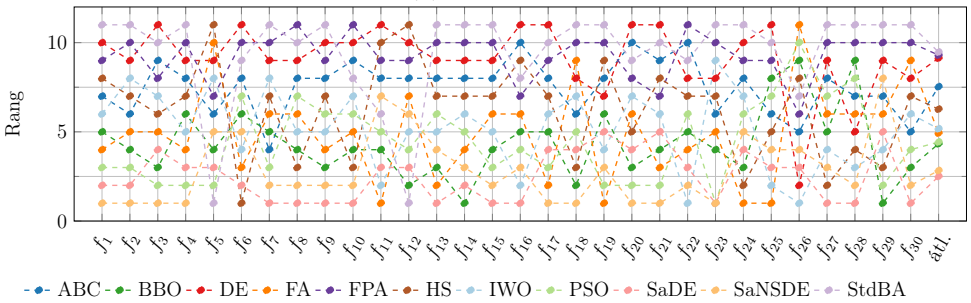
(a) $n_D = 2$



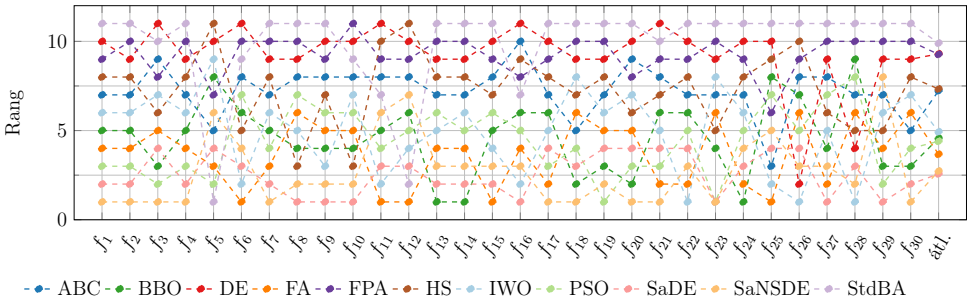
(b) $n_D = 10$



(c) $n_D = 30$



(d) $n_D = 50$



(e) $n_D = 100$

3.2. ábra. Az algoritmusok rangja különböző n_D értékek esetén

Az így kapott rangsort problémanagyságonkénti és függvényenkénti lebontásban algoritmusonként a 3.2. ábra szemlélteti, és az átlagos rangot összefoglalva a teljes tesztkészletre a 3.3. táblázat tartalmazza. Látható, hogy a legjobb eredményt adó, és egyben a jelen vizsgálati esetben a legmegbízhatóbb algoritmusok az önadaptivitással rendelkező SaDE és SaNSDE.

A 2.3 - 2.5 részekben bemutatott feladatok optimálásához az FA, FPA, és SaDE algoritmusokat használom fel. Igaz, hogy a 3.3. táblázat szerint az FA és FPA algoritmus nem tartozik a leghatékonyabb eljárások közé, de ez a kimutatás elkészítése időben hosszadalmas volt, és algoritmus választáskor még csak részeredményeim voltak.

3.3. táblázat. Algoritmusok átlagos rangja

n_D	Átlagos rang										
	ABC	BBO	DE	FA	FPA	HS	IWO	PSO	**SaDE**	**SaNSDE**	StdBA
2	9,00	9,64	3,50	7,77	3,09	7,00	4,36	6,14	**3,59**	**2,95**	5,14
10	8,80	6,90	4,57	7,37	8,90	5,40	5,20	6,30	**2,23**	**2,83**	7,00
30	7,77	5,60	8,20	4,80	9,47	5,67	5,20	5,10	**2,20**	**2,63**	9,20
50	7,53	4,33	9,13	4,93	9,30	6,27	5,17	4,43	**2,50**	**2,83**	9,47
100	7,23	4,57	9,30	3,67	9,27	7,33	4,93	4,40	**2,57**	**2,73**	9,90
átlag	8,07	6,21	6,94	5,71	8,00	6,33	4,97	5,27	**2,62**	**2,80**	8,14

4. fejezet

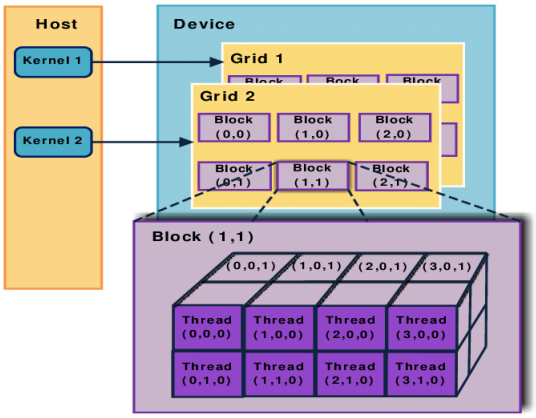
Evolúciós számítások párhuzamos feldolgozása

Az evolúciós algoritmussal történő optimálás során a célfüggvényérték kiszámítások száma 10^5 - 10^6 nagyságrendbe is eshet, vagy szélsőséges esetben még ennél is több. Könnyen belátható, hogy ugyanennyiszer kell új egyedet is generálni. Ez igen nagy számítási kapacitást igényel. (Csak érdekességképpen az előző fejezetben szereplő szimuláció lefuttatása algoritmusonként 6 – 10 napig tartott.)

Felmerül a kérdés, hogy időben csökkenthető-e egy ilyen optimalizáció? Igen, a párhuzamos feldolgozás segítségével. Napjainkban egy viszonylag olcsó párhuzamosítási lehetőséget kínálnak a grafikus kártyák általános számításokhoz történő felhasználása.

4.1. CUDA Runtime API általános számításokhoz

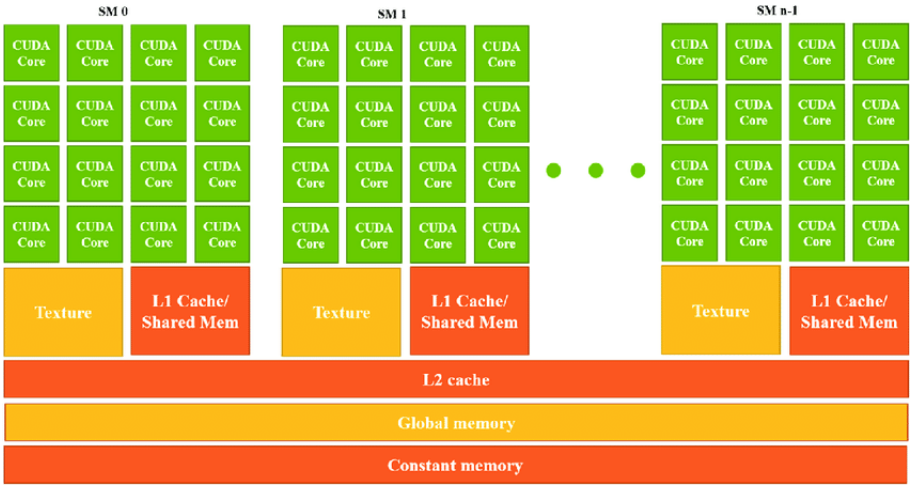
Az NVIDIA az általa gyártott grafikus kártyákhoz kínál egy általános célú alkalmazásprogramozási felületet (API-t) a *CUDA Driver API*-t és a *CUDA Runtime API*-t. Az első egy alacsony szintű interfész, és a programozása is nehézkes. Az utóbbi pedig egy magasabb absztrakciós szint, mely már könnyebben programozható.



4.1. ábra. Szálkezelés struktúrája. Forrás: [42]

CUDA-ban a GPU-n nagyszámú egymással párhuzamosan futó szálat (számítást) indíthatunk el strukturált formában (lásd 4.1. ábra). A szervezés legalsó szintjén találhatóak maguk a szálak (threads). A szálakat blokkokba (blocks) lehet szervezni. Egy blokkon belül a szálak indexelése három koordinátával történik. Gyakran szokás ezt egy térbeli kockával

szemléltetni. A blokkok rácsokba (grid) vannak szervezve, itt az indexelés már csak két koordinátával történik. Magukat a rácsokat pedig egy index azonosítja. Nem véletlen ez a szervezési mód. Jobb megértéshez érdemes megvizsgálni a GPU-k felépítését (4.2). ábra. Minden GPU több multiprocesszorból áll (lásd 4.2. ábra $SM_1, SM_2, \ldots, SM_{n-1}$), amiket ALU-k tömbje (CUDA core / CUDA mag) alkot. Az elindított rács blokkjait és annak szálait egy feladatütemező osztja szét a multiprocesszorok és a magok között. Programozói szinten a blokkok és a szálak kiosztását érdemes véletlenszerűnek tekinteni, mert nincs a felhasználónak ráhatása. Egy blokk addig marad egy multiprocesszorban, míg az összes szála le nem fut. Ha végzett a futással új blokk töltődik be és kezdi meg futását. Ennek következménye, hogy egy már lefutott blokkot nem lehet visszatölteni. Továbbá ebből következik az is, hogy a szálaknak egymástól függetlennek kell lenniük.



4.2. ábra. GPU szerkezete. Forrás: [117]

A szálak, blokkok és gazdagép közötti kommunikációt a különböző szinteken definiált memóriák biztosítják. Legalacsonyabb szinten a magokhoz köthető regiszterek vannak. Minden szál kap belőlük egy készletet, és csak az a szál írhatja, olvashatja, akinek a tulajdonában áll. A szálak közötti kommunikációt az osztott memória (*L1 cache/ Shared mem.*) biztosítja, ezt az adott multiprocesszoron futó blokk szálai érhetik el. A klasszikus memória szerepét a globális memória tölti be. Ezt a területet mind a multiprocesszorok, mind a gazdagép látja és írhatja. Továbbá létezik még konstans-, és textúramemória is, melyek szerepe itt nem kerül bemutatásra, leírásuk a [22] és az [51] irodalmakban megtalálható.

A CUDA három függvénytípust különböztet meg az alapján, hogy ki hívhatja és hol fut:

– *kernelfüggvények*: a GPU-n párhuzamosan futó kódok kerülnek ide. Csak a gazdagép hívhatja meg őket, és meg kell mondani, hogy hány blokkban, mennyi szálat kíván elindítani. Egy adott blokkon belül, a szálakban kvázi ugyanaz a programkód fut, de más-más adaton képesek dolgozni. Így kerül megvalósításra a SIMD párhuzamos programozási architektúra. Általános alakja a kernel függvényhívásnak:

```
függvénynév <<<blokkok,szálak>>> (függvényparaméterek);
```

– *eszközfüggvények*: GPU-n szintén párhuzamosan futni képes kisebb programozási egységek. Kizárólag a grafikus kártyán futó kód képes meghívni és egyszerre csak

egyet. Ebből következik, hogy kernel függvényből nem indítható további párhuzamos végrehajtás;

– *normál függvények*: lényegében a hagyományos programozási technikában megszokott a CPU-n egy szálon futó függvények. Csak a gazdagépen futó függvényekből hívhatóak. Több processzormagos, vagy időosztásos rendszereken akár több valódi vagy látszólagos szálon is képesek futni, de ez a témakör nem tartozik jelen dolgozat tárgykörébe. Szinkronizálhatóak a GPU eszközökön futó program részekkel. Vagy bevárják azt, vagy vele párhuzamosan futnak, vagy mindkét lehetőséget megvalósítva felügyeleti, vezérlési feladatokat látnak el.

4.2. Evolúciós algoritmusok párhuzamos végrehajtásának lehetőségei

A 2. fejezetben bemutatott 1. algoritmus magas absztrakciós szinten mutatja be az EA-k általános felépítését. Alacsonyabb absztrakciós szinten az általánosításban megjelennek az egyedekkel végzendő ismétlődő műveletek is, részletesebben a 8. és a 9. algoritmusok. Az első esetben (8. algoritmus) a rekombináció, mutáció és visszahelyezés műveletek függenek az előző egyeden elvégzett művelet eredményeitől is. Látható, hogy ezek egymás után egyedenként kerülnek meghatározásra. Ezt a felépítést követi például a FA algoritmus.

A másik lehetőség (9. algoritmus) pedig, hogy a műveleteket egyenként egymás után elvégezzük a teljes populációra. Mind a két megközelítés lehetőséget ad valamilyen fokú párhuzamosításra, de a 9. algoritmus szerinti evolúciós módszerek kínálnak nagyobb szabadságfokot. Tipikusan ilyen a DE, az IWO, és sok más algoritmus.

A [21] és [25] szakirodalmak a párhuzamosításnak három csoportját különböztetik meg:

– *globális modell*: a legegyszerűbb párhuzamosítási technika. Csak a különböző műveletek és számítások kerülnek párhuzamosításra. A populáció az eredeti formájában marad, nincs semmilyen strukturálás.

– *regionális modell*: A teljes populációt egyenlő nagyságú részpopulációkra bontjuk. Minden egyes részpopuláción ugyanaz az EA önállóan fut. Teljes izoláció elkerülése végett időnként a részpopulációk egyedet cserélnek egymással.

Algoritmus 8: Az EA általános felépítése, részeredményektől függő részfeladatokkal

```
1 (0)P populáció inicializálása
2 while kilépési feltétel hamis do
3     for i ∈ (G)P do
4         if szelekciós feltétel igaz i-re then
5             (G+1)P_p = (G+1)P_p ∪ i
6     for i ∈ (G)P do
7         (G+1)P_r = (G+1)P_r∪ rekombináció(i,(G+1) P_p,(G+1) P_r)
8         (G+1)P_m = (G+1)P_m∪ mutáció(i,(G+1) P_r,(G+1) P_m)
9         (G+1)P = (G+1)P∪ visszahelyezés(i,(G+1) P_m,(G+1) P)
```

Algoritmus 9: Az EA általános felépítése, részeredményektől nem függő részfeladatokkal

```
1  (0)P populáció inicializálása
2  while kilépési feltétel hamis do
3      for i ∈ (G)P do
4          if szelekciós feltétel igaz i-re then
5              (G+1)P_p = (G+1)P_p ∪ i
6      for i ∈ (G)P do
7          (G+1)P_r = (G+1)P_r ∪ rekombináció(i, (G+1)P_p)
8      for i ∈ (G)P do
9          (G+1)P_m = (G+1)P_m ∪ mutáció(i, (G+1)P_r)
10     for i ∈ (G)P do
11         (G+1)P = (G+1)P ∪ visszahelyezés(i, (G+1)P_m)
```

– *lokális modell*: abban az esetben alkalmazható, ha egy nagyon komplex feladat részmegoldások összességéből áll elő. Ekkor egy-egy részpopuláció más-más részfeladaton dolgozik. Időnként ezek az al-populációk „kooperálnak" egymással és összevetik, hogy a részfeladatok megoldásában elért eredmények a teljes egészet tekintve jobb eredményt adnak-e?

A GPU-n történő párhuzamosítás szempontjából a legjobban alkalmazható a globális modell és a 9. algoritmus szerinti felépítés. Ezek a megközelítések állnak a legközelebb a SIMD architektúrákhoz. A továbbiakban csak ezzel foglalkozom.

4.3. Virágbeporzási algoritmus párhuzamosítása

A teljes algoritmus a 9. algoritmus szerinti általánosított elvi felépítést követi, ezért jól párhuzamosítható. A (2.34) és (2.37) egyenletekben található véletlen számokat érdemes n_p méretű vektorokban tárolni

$$\mathbf{L} = [L_1, L_2, L_3, \ldots, L_{np}]^T \,, \tag{4.1}$$

$$\epsilon = [\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_{np}]^T \,, \tag{4.2}$$

$$\mathbf{j} = [j_1, j_2, j_3, \ldots, j_{np}]^T \,, \tag{4.3}$$

$$\mathbf{k} = [k_1, k_2, k_3, \ldots, k_{np}]^T \,, \tag{4.4}$$

$$\mathbf{p}_p = [p_{p,1}, p_{p,2}, p_{p,3}, \ldots, p_{p,np}]^T \,, \tag{4.5}$$

ahol $\mathbf{p}_p$ vektor elemei [0,1) félig zárt intervallumon vett egyenletes eloszlású véletlen számok. A rekombinációs művelet megkezdése előtt ezen vektorok elemei egymástól függetlenül generálhatóak n_p darab szálon, vektoronként külön blokkban. A G. generáció ${}^{(G)}\mathbf{x}_i$ egyedeit pedig érdemes egy ${}^{(G)}\mathbf{X}$ mátrixban tárolni

$${}^{(G)}\mathbf{X} = \begin{bmatrix} {}^{(G+1)}\mathbf{x}_1 \\ {}^{(G+1)}\mathbf{x}_2 \\ {}^{(G+1)}\mathbf{x}_3 \\ \vdots \\ {}^{(G+1)}\mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} {}^{(G+1)}X_{1,1} & {}^{(G+1)}X_{1,2} & {}^{(G+1)}X_{1,3} & \ldots & {}^{(G+1)}X_{1,D} \\ {}^{(G+1)}X_{2,1} & {}^{(G+1)}X_{2,2} & {}^{(G+1)}X_{2,3} & \ldots & {}^{(G+1)}X_{2,D} \\ {}^{(G+1)}X_{3,1} & {}^{(G+1)}X_{3,2} & {}^{(G+1)}X_{3,3} & \ldots & {}^{(G+1)}X_{3,D} \\ \vdots & & & \ddots & \vdots \\ {}^{(G+1)}X_{np,1} & {}^{(G+1)}X_{np,2} & {}^{(G+1)}X_{np,3} & \ldots & {}^{(G+1)}X_{np,nD} \end{bmatrix} , \tag{4.6}$$

```
__global__ void recombinationFPA(double *X, double *newX,
        double *L, double *epsilon, int *j, int *k, double *r, int iBest,
        double pp) {

                //Párhuzamos környezet azonosítása
                int a=threadId.y; int b=threadId.x;

                //Szükséges indexek meghatározása
                int id=blockDim.x*a+b;
                int bestId=blocDim.x*iBest+b;
                int jId=blockDim.x*j[a]+b;
                int kId=blockDim.x*k[a]+b;

                //Rekombinációs művelet
                if (pp<=r[a]) {
                        newX[id]=X[id]+L[a]*(X[id]-X[bestId]);
                } else {
                        newX[id]=X[id]+epsilon[a]*(X[jId]-X[kId]);
                }
        }
```

4.1. Forráskód. Párhuzamos FPA rekombinációs műveletének forráskódja (részlet)

ahol a sorok jelentenek egy-egy egyedet, oszlopok pedig az egyedekhez tartozó koordinátákat.

Az $\mathbf{X}$ egy-egy $X_{a,b}$ eleme a mátrix többi elemétől függetlenül kiszámolható a (2.34), (2.37) és (2.38) egyenletek összevonásával

$$ {}^{(G+1)}X_{a,b} = \begin{cases} {}^{(G)}X_{a,b} + L_a \left({}^{(G)}X_{a,b} - {}^{(G)}X_{best,b} \right) & \text{ha } p_p \leq p_{p,i} \\ {}^{(G)}X_{a,b} + \epsilon_a \left({}^{(G)}X_{j_a,b} - {}^{(G)}X_{k_a,b} \right) & \text{egyébként} \end{cases} . \tag{4.7} $$

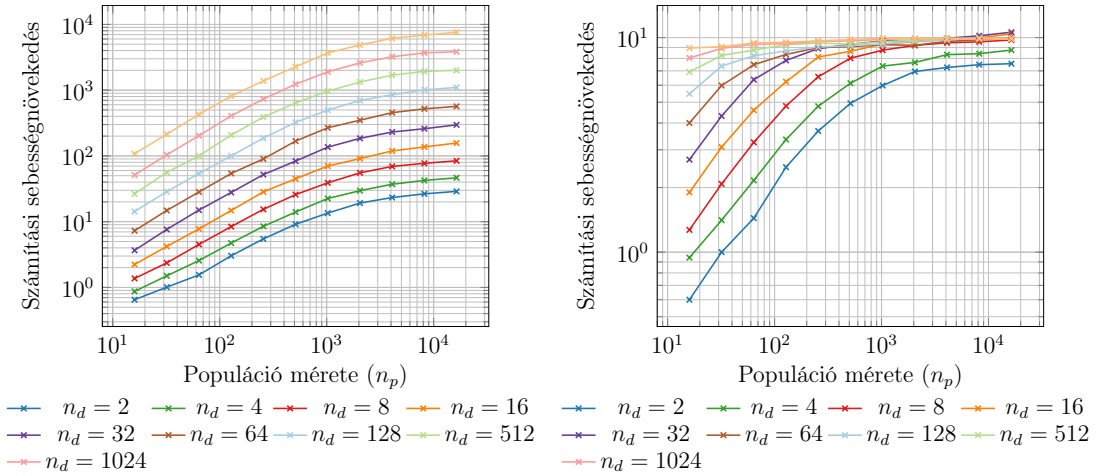
A (4.7) egyenlet alapján 4.1. forráskód szerint $\mathbf{X}$ mátrix elemeinek megfelelő szál indítható. A szálakat 2D tömbbe kell szervezni, mely kialakítása teljes mértékben megegyezik $\mathbf{X}$ alakjával. Észrevehető, hogy a 4.1. forráskód nem „hibatűrő”, mert ha a `blockDim.x` nem egyezik meg n_D-vel, a `blockDim.y` pedig n_p-vel, akkor nagyon könnyen lehet rossz memóriaterületre hivatkozni. Ez ellen védekeznie a kódot felhasználó programnak kell, magában a `__global__` függvényben nincs rá lehetőség.

Az FPA visszahelyezés művelete

$$ {}^{(G+1)}\mathbf{x}_i = \begin{cases} {}^{(G+1)}\mathbf{x}_i & \text{ha } \mathcal{F}\left({}^{(G+1)}\mathbf{x}_i \right) < \mathcal{F}\left({}^{(G)}\mathbf{x}_i \right) \\ {}^{(G)}\mathbf{x}_i & \text{egyebként} \end{cases} \tag{4.8} $$

alapján definiálható. Itt a párhuzamosítás lehetőségét csak abban az esetben kell megfontolni, ha maga a fitneszfüggvény is párhuzamosan kerül kiszámításra. Ellenkező esetben fölösleges memóriatartalom másolással szembesülünk. Ha a fitneszérték is párhuzamosan számítódik, érdemes azt az összes egyedre még a szelekciós művelet előtt, előre kiszámolni, nem pedig a szelekció közben.

Egy iterációs lépés elvégzéséhez szükséges idő két részből tevődik össze. Egyrészt t_{alg} optimalizációs algoritmus műveleteinek az idejéből, másrészt a t_{fgv} függvény érték(ek)



(a) FPA műveleteknél elért számítási sebességnövekedés

(b) Teljes iterációs lépés során elért számítási sebességnövekedés

4.3. ábra. Párhuzamos futtatással elért eredmények

kiszámításához szükséges időből

$$t_{it} = t_{alg} + t_{fgv}\,, \qquad \tilde{t}_{it} = \tilde{t}_{alg} + t_{fgv}\,, \tag{4.9}$$

ahol t_{it} egy iterációs lépés ideje, $\tilde{t}$ pedig a párhuzamosan GPU-n futó lépés idejét jelöli. Az FPA algoritmus párhuzamos futtatásával $t_{alg}/\tilde{t}_{alg}$ hányadosként értelmezett sebesség növekedést szemlélteti a 4.3a. ábra. Nagy n_p egyedszám és n_d sokdimenziós feladat esetén akár $\sim 10^4$-szer is futhat gyorsabban. Nagyon kicsi n_p és n_D esetén akár lassabb is lehet a hagyományos szekvenciális futásnál. A párhuzamos futás előkészítése és magának a műveletnek az elvégzése több időt igényel, mint normál körülmények között annak elvégzése.

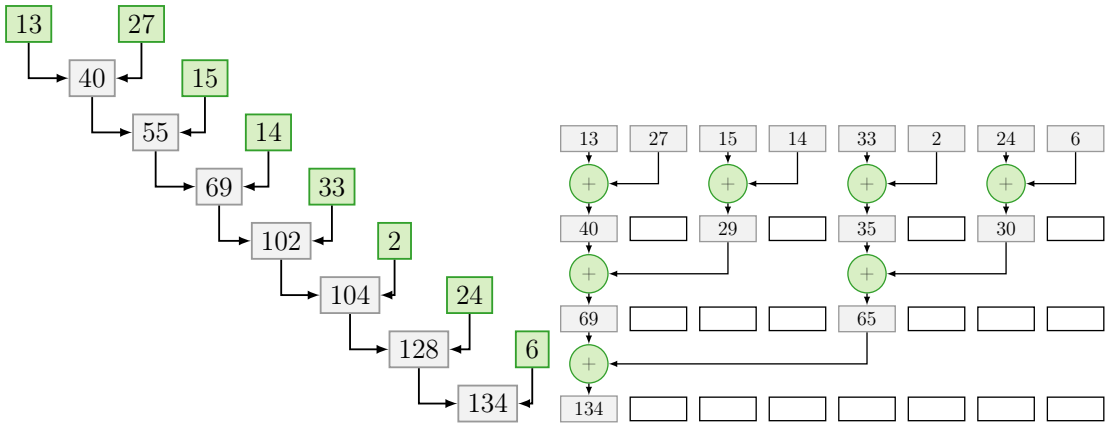
A 4.3b. ábra szemlélteti azt az esetet, mikor a teljes iterációs lépést figyeljük és a számítási sebesség növekedést $t_{it}/\tilde{t}_{it}$ hányadossal definiáljuk. A szimulációhoz nagyon egyszerű

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{n_d} x_i^2 \tag{4.10}$$

függvényt választottam. A függvényérték kiszámítása szekvenciálisan történik. Az így elérhető sebesség növekedés kb. 10^1, mivel a feladat nagyságának növekedésével egyre jobban a függvényérték meghatározásához szükséges idő dominál.

4.4. Fitneszfüggvények párhuzamos számítása

Szekvenciális feldolgozás során a matematikai és/vagy logikai kifejezések kétoperandusú műveletek egymás utáni meghatározott sorrendű elvégzéséből áll. Ezt a folyamatot szemlélteti egy $n = 8$ tagú számsorozat összegzésére a 4.4a. ábra. A kifejezés kétoperandusú (bináris) műveletek sorozatára kerül felbontásra, az első operandus az előző művelet eredménye, a második operandus pedig egy eleme az eredeti kifejezés tagjainak. Nem homogén műveletekből felépülő kifejezés precedencia szabályt betartó felbontás a *Shunting-yard* [106] algoritmussal elvégezhető rekurzívan. Szekvenciálisan a kiértékelés műveletigénye $O(n-1)$, ahol n az operandusok száma.



(a) Szekvenciális feldogozás (b) Párhuzamos feldolgozás

4.4. ábra. Számsorozat összegezése

A kifejezések párhuzamos kiértékelésére létezik a párhuzamos-redukció algoritmus [45, 100]. Az eredeti algoritmussal csak homogén kétoperandusú műveletekből felépülő kifejezések értékelhetőek ki. Egy-egy operanduspárra páronként egyszerre kiértékelhető a művelet, majd következő lépésben az így kapott eredményekkel páronként ismét egyszerre elvégezhető a művelet. Ezt a lépéssorozatot addig kell ismételni, míg végül megkapjuk a végeredményt. A 4.4a. ábrán látható szekvenciális összegzés párhuzamos redukcióval történő kiértékelési folyamatát szemlélteti a 4.4b. ábra. Ha az operandusok száma 2-nek nem egész hatványa, akkor szükséges azokat úgy kiegészíteni 2 egész hatványára, hogy az a végeredményt ne befolyásolja. Az implementációs technikák a homogenitásra vonatkozó követelményt nem oldják fel. A párhuzamos kiértékelés műveletigénye $O\left(\log_2 n\right)$.

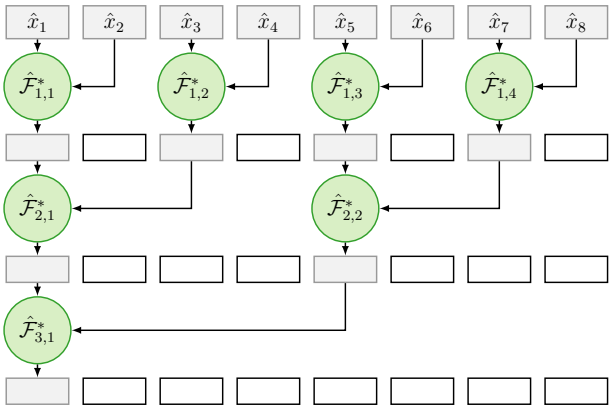
A továbbiakban tekintsünk függvénynek minden matematikai értelemben vett függvényt és minden kétoperandusú műveletet (pl.: összeadás, kivonás, szorzás, osztás stb.). Ekkor $\hat{\mathcal{F}}(\hat{\mathbf{x}})$ jelölje az $\mathcal{F}(\mathbf{x})$ eredeti fitneszfüggvény hierarchikus felbontását. $\hat{\mathcal{F}}(\hat{\mathbf{x}})$ csak $\hat{\mathcal{F}}_i(\hat{\mathbf{x}}_i)$ egyszerű függvényekből épülhet fel. $\hat{\mathcal{F}}_i(\hat{\mathbf{x}}_i)$ olyan kétváltozós függvény, ahol a változók között értelmezett kifejezés kiértékelésében a szekvencialitás nem kerülhető el, vagy nem érdemes annak elkerülésével a komplexitást növelni. A kifejezés tartalmazhat konstanst is, mely feldolgozásának párhuzamos lehetőségeitől érdemes eltekinteni, jelen esetben. Például ilyen egyszerű függvény lehet

$$\hat{\mathcal{F}}_1(a,b) = a + b\,, \quad \hat{\mathcal{F}}_2(a,b) = a \cdot b + c\,, \quad \hat{\mathcal{F}}_3(a,b) = c\sqrt{(a+b)}\,, \quad \hat{\mathcal{F}}_4(a,b) = a\,, \tag{4.11}$$

ahol a és b függvényváltozók, c egy konstans. Az $\hat{\mathbf{x}}$ vektor az eredeti $\mathbf{x}$ vektor elemeiből felépített, azok variációját tartalmazó vektor. Szükség esetén helyenként feltöltve a végeredményt nem befolyásoló konstansokkal.

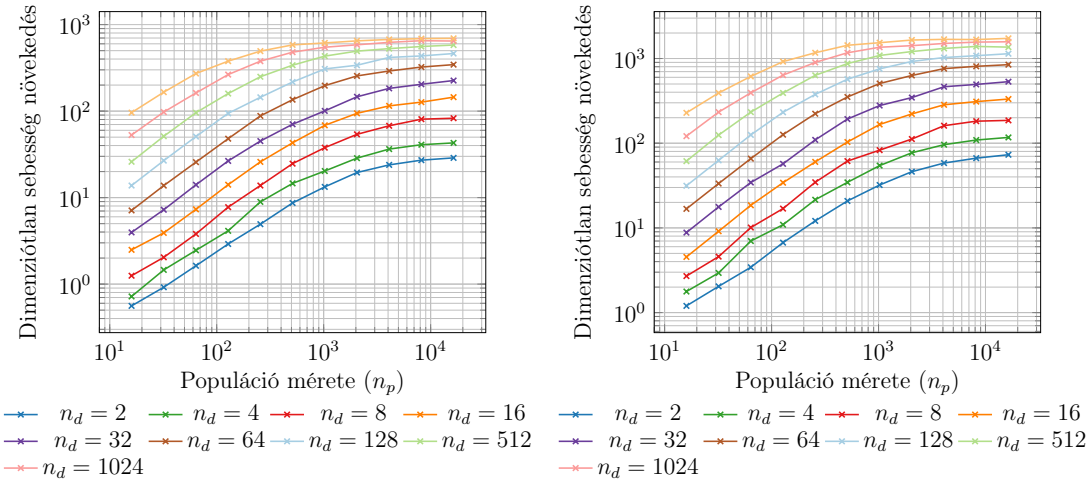
A $\mathcal{F}(\mathbf{x})$ függvény $\hat{\mathcal{F}}(\hat{\mathbf{x}})$ szerinti hierarchikus felbontását szemlélteti a 4.5. ábra. Az $\hat{\mathcal{F}}(\hat{\mathbf{x}})$ függvényt alkotó $\hat{\mathcal{F}}_{i,j}(\hat{\mathbf{x}}_{i,j})$ függvények címét (*pointer*-ét) jelölje $\hat{\mathcal{F}}^*_{i,j}$. A függvényekhez tartozó címekből a hierarchiának megfelelően építhető egy mátrix

$$\mathbf{O}_p = \begin{bmatrix} \hat{\mathcal{F}}^*_{1,1} & \hat{\mathcal{F}}^*_{1,2} & \hat{\mathcal{F}}^*_{1,3} & \hat{\mathcal{F}}^*_{1,4} & \cdots & \hat{\mathcal{F}}^*_{1,n} \\ \hat{\mathcal{F}}^*_{2,1} & \hat{\mathcal{F}}^*_{2,2} & \hat{\mathcal{F}}^*_{2,3} & \cdots & \hat{\mathcal{F}}^*_{1,n/2} & null \\ \vdots & & & & & \\ \hat{\mathcal{F}}^*_{m-1,1} & \hat{\mathcal{F}}^*_{m-1,n/2^{m-2}} & null & null & \cdots & null \\ \hat{\mathcal{F}}^*_{m,1} & null & null & null & \cdots & null \end{bmatrix}, \tag{4.12}$$



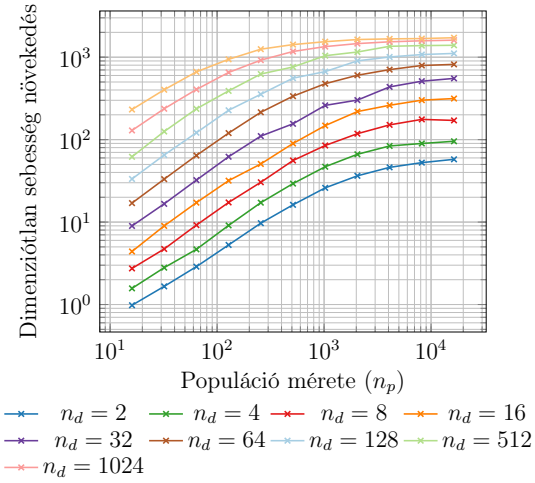
4.5. ábra. $\hat{\mathcal{F}}$ függvény hierarchikus ábrázolása

ahol a *null* értékek *null pointert*-t jelölnek. A párhuzamos redukcióhoz szükséges homogén művelet pedig legyen az $\mathbf{O}_p$ mátrix elemeinek segítségével mutatón keresztüli függvényhívás, ezt szemlélteti a 4.5. ábra zöld körei. A gondolatmenettel sikerült egy inhomogén



(a) Sphere függvény esetén

(b) Ackley's függvény esetén



(c) Rastrigin függvény esetén

4.6. ábra. Teljesen párhuzamosan futó optimalizálással elérhető sebesség növekedések

```
__global__ void cudaAckley(double *g_iData,
double *g_oData, unsigned int n) {

//Szükséges változók definiciója
extern __shared__ double sumAckley[];
unsigned int tid = threadIdx.x;
unsigned int i = blockIdx.x*blockDim.x+threadIdx.x;

//Másolás global memory-ból shared memory-ba, és közben az
//inhomogenitást okozó műveletek elvégzése
sumAckley[tid]= (i < n) ? g_iData[i]*g_iData[i] : 0;
sumAckley[tid+n]= (i < n) ? cos(2 * M_PI * g_iData[i]) : 0;
__syncthreads();

//Maga a párhuzamos-redukció, összegzéssel
for (unsigned int s=blockDim.x/2; s>0; s>>=1) {
if (tid < s) {
sumAckley[tid]+=sumAckley[tid+s];
sumAckley[tid+n]+=sumAckley[tid+n+s];
}
__syncthreads();
}

//Hierarchia utolsó sorában lévő művelet ismét különbözik a
// többitől
if (tid==0) {
g_oData[blockIdx.x]=(-20.0)*exp((-0.2)*sqrt(sumAckley[0]/n))
-exp(sumAckley[n]/n)+20+exp(1.0);
}
}
```

4.2. Forráskód. Ackley's függvény párhuzamos kiszámításának forráskódja (részlet)

műveleteket tartalmazó kifejezést kiértékeltetni párhuzamos redukcióval. Ha a műveletekbeli inhomogenitást a hierarchia első sora tartalmazza csak, akkor az O_p mátrix elhagyható. Ekkor ezek a műveletek egyidőben elvégezhetőek a szükségszerű *global memory*-ból *shared memory*-ba történő másolással, mint ahogyan azt a 4.2. forráskód is szemlélteti az *Ackley's* függvény esetén [67]. Igaz, esetenként körültekintőbb kódolást igényelhet, és sokkal komplexebb kódot eredményezhet, de jóval nagyobb számítási teljesítmény érhető így el, mintha *pointer*-en keresztüli függvényhívásokat indítunk.

Az evolúciós algoritmusok fitneszfüggvényét általában a teljes populációra ki kell számolni, ezért egyetlen egy *grid*-ben az $\mathbf{O}_p$ méretével megegyező *block* mátrixban célszerű elindítani a párhuzamos számítást. A populáció egyes egyedeihez köthető számításokat pedig a *block*-on belüli szálak jelentsék. Természetesen ez a megközelítés csak addig alkalmazható, míg a feladat nagysága el nem éri a grafikus kártya memóriakapacitásának határait. A dolgozat keretein belül nem került megvizsgálásra, hogyan célszerű felbontani a számítást abban az esetben, ha a feladat nagysága elérte vagy túllépte a GPU memóriakapacitását.

[67]-ben definiált három tesztfüggvényre, úgymint *Sphere*, *Ackley's* és *Rastrigin* az optimáló algoritmus és magának a fitneszfüggvény párhuzamos feldolgozásával elérhető

$$\hat{t} = \frac{t_{alg} + t_{fgv}}{\tilde{t}_{alg} + \tilde{t}_{fgv}} \tag{4.13}$$

hányadossal definiált sebessé növekedést szemlélteti a 4.6. ábra. Ezek az eredmények sokkal jobbak, mint a 4.3b. ábrán szemléltetettek. Nagy populáció és sokdimenziós feladat esetén ez akár 10^3 nagyságrendbe eshet. Itt is meg kell jegyezni, kis méretű feladatok esetén semmi értelme nincs a párhuzamosításnak, mert a környezet előkészítéséhez befektetett műveletek elvégzése sokkal több időt igényel, mint amit a párhuzamos futással lehet nyerni.

A módszer validációjához nem elegendő csak tesztfüggvények esetén elvégezni a teljes optimálása párhuzamosítását. A 2.4. részben bemutatásra kerülő futódaru főtartójának optimálásához szükséges fitneszfüggvény párhuzamos számítási modelljét is elkészítettem. Terjedelmi okok miatt csak a 2.4. részben az (2.98) egyenlettel definiált T_3 hegesztési idő párhuzamos kiértékeléséhez szükséges modellt ismertetem. Az összes többi esetben is hasonlóan kell eljárni.

A (2.98) egyenletben szereplő összes mennyiséget kifejezve az optimálás változóival, és zárójelek felbontása után az alábbi formát kapjuk

$$T_3 = \Theta_3 \sqrt{\kappa_3 \rho L h t_w + \kappa_3 \rho L b t_f + 10{,}0625 \kappa_3 \rho t_s b h} + 3 \cdot 1{,}3 \cdot 0{,}3394 \cdot 10^{-3} \cdot 0{,}5^2 t_f^2 L\,, \tag{4.14}$$

ahol Θ_3, κ_3, ρ, L, és t_s mennyiségek az iterációs lépések során konstansnak tekinthetők. Az egyenletben azonosítható $\hat{\mathcal{F}}_i$ egyszerű függvények

$$\hat{\mathcal{F}}_1(\hat{x}_1, \hat{x}_2) = \tilde{\mathcal{F}}_1 \hat{x}_1 \hat{x}_2\,, \qquad \tilde{\mathcal{F}}_1 = \kappa_3 \rho L\,, \tag{4.15}$$

$$\hat{\mathcal{F}}_2(\hat{x}_1, \hat{x}_2) = \tilde{\mathcal{F}}_2 \hat{x}_1 \hat{x}_2\,, \qquad \tilde{\mathcal{F}}_2 = 10{,}0625 \kappa_3 \rho t_s\,, \tag{4.16}$$

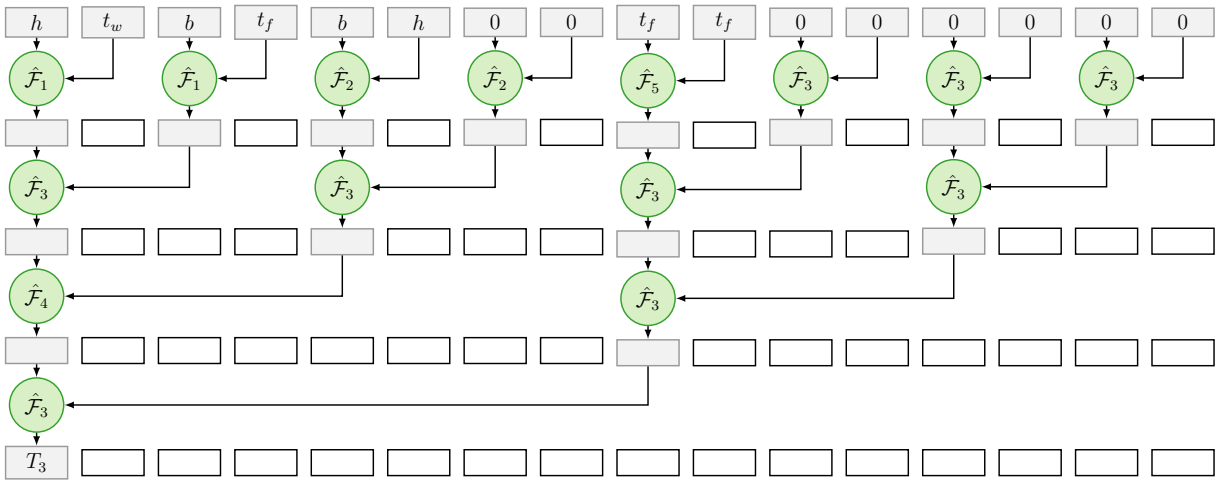
$$\hat{\mathcal{F}}_3(\hat{x}_1, \hat{x}_2) = \hat{x}_1 + \hat{x}_2\,, \tag{4.17}$$

$$\hat{\mathcal{F}}_4(\hat{x}_1, \hat{x}_2) = \tilde{\mathcal{F}}_4 \sqrt{\hat{x}_1 + \hat{x}_2}\,, \qquad \tilde{\mathcal{F}}_4 = \Theta_3\,, \tag{4.18}$$

$$\hat{\mathcal{F}}_5(\hat{x}_1, \hat{x}_2) = \tilde{\mathcal{F}}_5 \hat{x}_1 \hat{x}_2\,, \qquad \tilde{\mathcal{F}}_5 = 3 \cdot 1{,}3 \cdot 0{,}3394 \cdot 10^{-3} \cdot 0{,}5^2 L\,. \tag{4.19}$$

A $\hat{\mathcal{F}}_1$ – $\hat{\mathcal{F}}_5$ felhasználásával megalkotható a $\hat{T}_3(\hat{\mathbf{x}}_3)$ függvény, melyet a 4.7. ábra szemléltet, ahol az eredeti

$$\mathbf{x} = \begin{bmatrix} h & t_w & b & t_f \end{bmatrix}^T \tag{4.20}$$

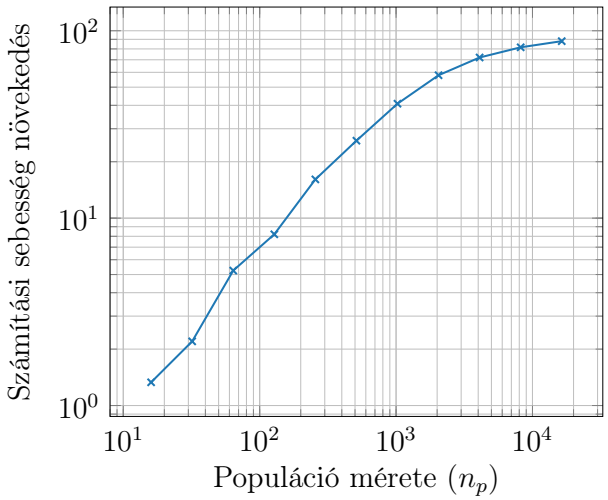


4.7. ábra. $\hat{T}_3(\hat{\mathbf{x}}_3)$ függvény hierarchikus ábrázolása

vektor elemeiből felépített, és méretre vonatkozó kritérium miatt a végeredményt nem befolyásoló elemekkel feltöltött vektor pedig

$$\hat{\mathbf{x}}_3 = \begin{bmatrix} h & t_w & b & t_f & b & h & 0 & 0 & t_f & t_f & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T . \tag{4.21}$$

Látható a 4.7. ábrán, hogy a T_3 idő meghatározásához elegendő négy egymás utáni lépés. Hasonló eredmények érhetőek el a 2.4. részben a (2.92) – (2.103) között definiált többi gyártási idővel, a (2.105) – (2.125) közötti korlátokkal, végül a (2.104) költség és (2.127) fitneszfüggvényekkel. Figyelembe véve azt is, hogy a teljes populációra ezen értékek kvázi egyidőben kerülnek meghatározásra, jelentős sebesség növekedés érhető el a (4.13) szerint, amit a 4.8 ábra szemléltet. Populáció mérettől függően akár 10^2-szor gyorsabban futhat az optimálás. Kis populáció méret ($n_p = 16$ vagy $n_p = 32$) esetén az előzőekhez hasonlóan sebesség csökkenés nem volt tapasztalható, az elért növekedés elhanyagolhatóan kicsi.



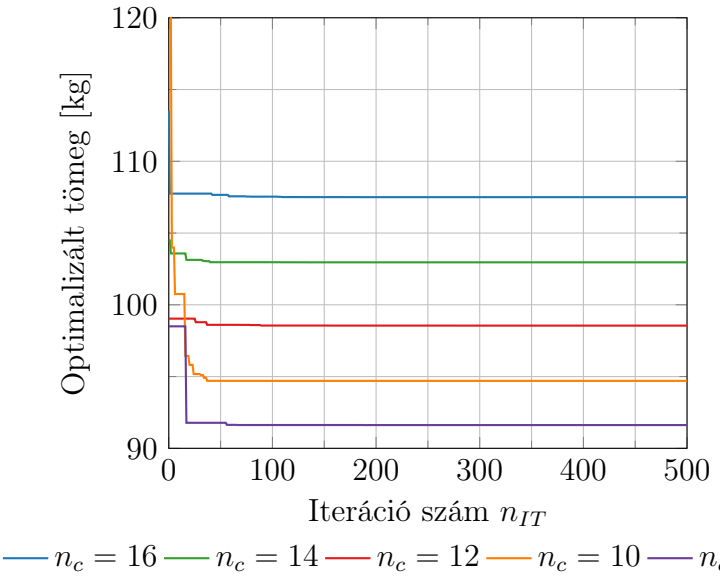
4.8. ábra. Számítási sebesség növekedés a futódaru főtartójának optimálása során

5. fejezet

Optimálások eredményei

5.1. Teherautó plató kereszttartó optimálásának eredményei

Az optimalizálást FPA algoritmussal végeztem, amihez a szükséges programokat[5] MatLAB-ban írtam meg. Megvizsgáltam több $n_c = 8 - 16$ kereszttartó esetén is, hogyan változik az optimalizált tartó keresztmetszetéhez tartozó tömeg. Minden esetben az alkalmazott populációméret $n_p = 100$ volt, a valószínűségi változót $p_p = 0{,}5$-re választottam, alkalmazott iterációszám n_{IT}. Az 5.1. ábrán látható, hogy a választott iterációszám sok, mivel körülbelül a 100. iterációs lépésre befejeződött a tényleges optimálás.



5.1. ábra. Kereszttartók optimálásának konvergenciája eltérő n_c kereszttartó szám esetén.

Kiszámoltam az anyagköltséget is $k_m = 1{,}72\,\$/\text{kg}$ fajlagos alapanyagköltséggel számolva [50]

$$K_m = k_m \rho_{Al} A_c L_{ct} n_c \,. \tag{5.1}$$

A gerinclemez vastagságát $t_w = 3{,}4\,\text{mm}$-re választottam a gyárthatósági kritériumokat figyelembe véve. A platólemez vastagságát $t_p = 2\,\text{mm}$-re választottam, ami végig állandó volt.

[5] A forráskód megtalálható a mellékelt adathordozón `./4_sz_melleklet/` könyvtárban

Az eredményeket az 5.1. táblázat foglalja össze, összehasonlítva az [50]-ben megjelent RHS keresztmetszet optimálás eredményeivel. Itt a szerzők *Hillclimb* algoritmust használtak.

5.1. táblázat. Optimálás eredményeinek összefoglalása

	Szelvény típusa							
	RHS [50]			I				
n_c	14,00	12,00	10,00	16,00	14,00	12,00	10,00	8,00
b [mm]	55,00	115,00	120,00	73,90	66,10	80,80	78,10	74,30
t_f [mm]	5,40	3,00	3,40	4,60	5,90	5,60	7,00	9,40
A_c [mm^2]	1274,00	1370,00	1496,00	1019,88	1116,36	1246,57	1437,52	1738,32
m_c [kg]	117,50	108,31	98,56	107,50	102,96	98,55	94,70	91,62
K_m [$]	202,10	186,28	169,51	184,90	177,09	169,50	162,89	157,58

Az I-szelvény minden esetben jobb eredményt adott mint az RHS. Az optimált tömeg és ez alapján a kereszttartók anyagköltsége is csökken a tartók számának csökkenésével. Ezért az [50]-ben javasolt minimális $n_c = 10$ értéket érdemes tovább csökkenteni $n_c = 8$ értékre.

5.2. Futódaru-főtartó optimálásának eredményei

Az optimalizálás a [121, 119] irodalmakban megjelent eredeti, alap szentjánosbogár (FA) algoritmussal került elvégzésre MatLAB programokkal[6], minden esetben $n_{IT} = 100$ iterációs lépéssel és $n_p = 100$ populáció mérettel. Az optimáló algoritmus további beállításait, és az alkalmazott paramétereket az 5.2. táblázat foglalja össze.

5.2. táblázat. FA algoritmus paraméterei futódaru-főtartójának optimálásához

Paraméter neve	jele	értéke
véletlen mozgás konstansa	α_l	0,20
véletlen mozgást csökkentő konstans	α_{damp}	0,95
vonzódási konstans	β_l	2,00
fényelnyelődési tényező	γ_l	1,00

A számszerűsített példában az alapértelmezett hasznos horog teher $P_h = 200\,\mathrm{kN}$, fesztáv $L = 16{,}5\,\mathrm{m}$ és a terhelési ciklusok száma $N = 2 \cdot 10^6$ ciklus. Megvizsgáltam, hogyan alakul az optimális költség, ha a hasznos horogteher $P_h = 50\,\mathrm{kN} - 500\,\mathrm{kN}$, a fesztáv $L = = 10\,\mathrm{m} - 40\,\mathrm{m}$ vagy pedig a terhelési ciklusok száma $N = 10^5 - 8 \cdot 10^8$ értékek között változik, a másik kettő érték pedig alapértelmezett marad. Az összehasonlítások során vizsgáltam még a különböző anyagminőségek hatását is. A konkrét anyagminőségeket az 5.3. táblázat foglalja össze.

[6] A forráskód megtalálható a mellékelt adathordozón `./5_sz_melleklet/` könyvtárban

A (2.104) egyenletben szereplő k_w műhelyköltséget egységnyire vettem fel, $k_w = 1\$/\text{min}$. Az anyagköltségeknél $f_{y1} = 235\,\text{MPa}$ folyáshatárú acélt tekintettem egységnyinek $k_m = 1\,\$/\text{kg}$. A többi nagyobb folyáshatárú acélnál pedig a tényleges árak közötti százalékos különbség került alkalmazásra. Részletesebben lásd az 5.3. táblázatban.

5.3. táblázat. Különböző acélminőségek árai

Ár (k_m)	Acél minőség			
	S235	S355	S460	S690
$\frac{\$}{\text{kg}}$	1	1,04	1,12	1,25

A bemutatásra kerülő optimalizálások során ugyanazon beállítások mellett több, egymástól független optimalizálást is futtattam. A 5.4 táblázat foglalja össze ötven független futás statisztikai eredményeit változó horogterhelés mellett. Látható, hogy az optimalizáció stabil eredményeket ad, mivel a kapott eredmények szórása nagyon kicsi. A többi vizsgált esetben is hasonlóan kis szórás tapasztalható. Az eredményeket globális minimumként kezelem.

5.4. táblázat. Változó horogterhelés mellett ötven egymástól független optimálás eredményének statisztikai adatai ($fy = 235\,\text{MPa}$, $L = 16{,}5\,\text{m}$ és $N = 2 \cdot 10^6$)

Horogteher [kN]	Költség [$]				
	min.	átl.	medián	max.	szórás
50	4379,54	4379,68	4379,69	4379,80	0,0532
100	5769,42	5769,58	5769,58	5769,74	0,0626
150	7129,79	7129,88	7129,88	7129,99	0,0448
200	8399,55	8399,69	8399,69	8399,85	0,0648
250	9591,63	9591,75	9591,75	9591,88	0,0593
300	10746,12	10746,27	10746,27	10746,49	0,0779
350	11904,50	11904,69	11904,69	11904,93	0,0906
400	13059,03	13059,18	13059,18	13059,33	0,0704
450	14218,54	14218,74	14218,73	14218,92	0,0812
500	15390,83	15391,01	15391,01	15391,26	0,0984

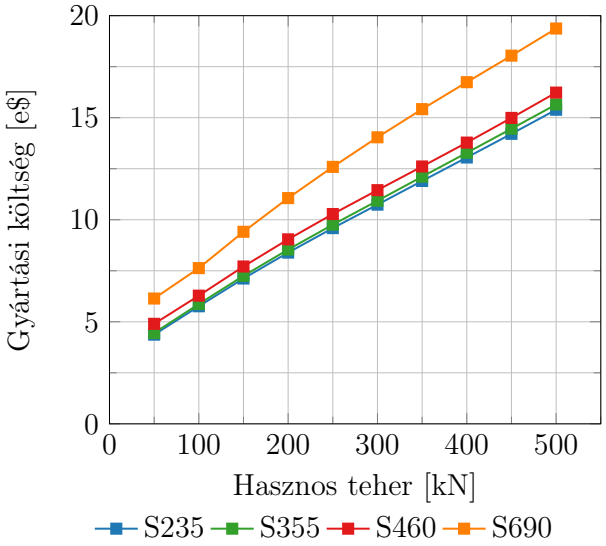
Változó horogteher mellett az optimalizált költségeket az 5.2. ábra szemlélteti, és a hozzájuk tartozó tervezési változó értékeket az 5.5. táblázat foglalja össze. Terhelésnövekedés hatására a költségminimum lineárisan növekszik. A kapott eredmények jó pontossággal közelíthetők egy

$$\hat{K}_1 = \hat{C}_1 \cdot P + \hat{C}_2 \tag{5.2}$$

egyenlettel, ahol $\hat{C}_1$, $\hat{C}_2$ konstansok értékeit különböző acélminőségekre az 5.6. táblázat foglalja össze. Az 5.6 táblázatból jól látható, hogy a közelítés nagyon kis hibát tartalmaz.

Az áthidalandó fesztáv változtatásával kapott eredményeket az 5.3. ábra szemlélteti, és a tervezési változók értékét az 5.7. táblázat foglalja össze. A költségváltozás a fesztáv függvényében közelíthető

$$\hat{K}_2 = \hat{C}_3 \cdot L^{\hat{C}_4} + \hat{C}_5 \tag{5.3}$$



5.2. ábra. Gyártási költség változása a hasznos teher függvényében

5.5. táblázat. Tervezési változók értéke változó horogterhelés mellett

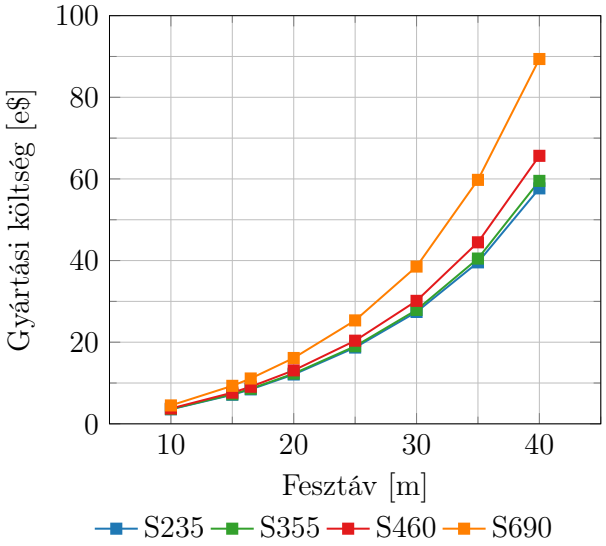
Horogteher [kN]	Tervezési változók [mm]															
	S235				S355				S460				S690			
	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f
50	533	320	18	8	526	315	18	9	515	309	18	10	493	296	22	12
100	558	391	19	14	559	391	19	14	536	375	19	16	514	360	23	16
150	597	418	20	18	600	419	20	18	602	421	21	17	552	387	24	21
200	645	451	22	20	648	453	22	19	654	456	23	18	601	421	26	22
250	687	481	23	21	690	482	23	21	699	489	24	19	642	449	28	24
300	743	520	25	20	747	523	25	20	742	518	25	20	681	476	29	25
350	781	546	26	21	784	549	26	21	800	560	27	19	714	500	31	26
400	816	571	27	22	818	572	27	22	831	582	28	21	746	522	32	27
450	848	594	28	23	851	595	28	23	856	599	29	22	791	554	33	25
500	879	616	30	24	883	618	30	24	887	620	30	23	821	575	34	26

egyenlettel, ahol a $\hat{C}_3$, $\hat{C}_4$ és $\hat{C}_5$ paraméterek értékeit az 5.8. táblázat tartalmazza. A $\hat{C}_4$ hatványkitevő ismeretében kijelenthető, hogy a kapcsolat jó közelítéssel köbös.

A terhelési ciklusuk változtatásával a költségfüggvény jellegében leköveti a kifáradási görbét, az eredmények az 5.4. ábrán láthatóak. Körülbelül $N = 4 \cdot 10^5$ ciklusszámig

5.6. táblázat. Terhelés - költség közelítőformula adatai

#	Acél minősége	Konstansok		Illeszkedési helyesség		Eltérés (RMSE)
		$\hat{C}_1$	$\hat{C}_2$	R^2	korr. R^2	
1.	S235	24,19	3407	0,9989	0,9987	130,88$
2.	S355	24,60	3469	0,9988	0,9987	134,17$
3.	S460	24,87	3895	0,9987	0,9985	144,34$
4.	S690	29,47	4947	0,9973	0,9969	247,86$



5.3. ábra. Gyártási költség változása a fesztáv függvényében

5.7. táblázat. Tervezési változók értéke változó fesztáv mellett

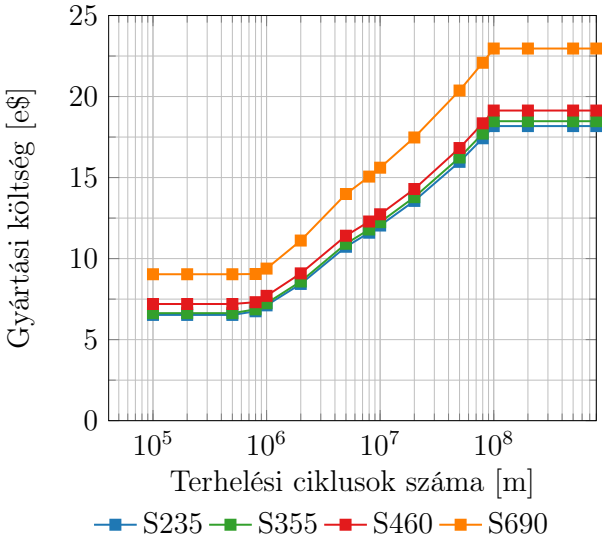
Fesztáv [m]	Tervezési változók [mm]															
	S235				S355				S460				S690			
	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f
10,0	497	348	17	17	498	348	17	17	504	342	17	17	467	308	20	21
15,0	612	428	21	20	614	430	21	19	624	427	21	18	571	399	25	22
16,5	646	452	22	20	648	453	22	20	656	458	23	18	603	422	26	22
20,0	737	516	25	19	738	517	25	19	732	512	25	19	675	472	30	23
25,0	864	605	29	18	864	605	29	18	845	591	30	19	789	552	35	22
30,0	982	687	33	18	968	678	32	19	944	661	33	21	937	562	41	23
35,0	1145	720	38	17	1134	680	38	19	1117	670	39	21	1079	647	48	26
40,0	1310	786	44	18	1287	772	43	22	1270	762	45	24	1232	739	54	29

a (2.119) egyenlet szerint meghatározott $\Delta\sigma_N$ feszültség értéke meghaladja az anyagra jellemző f_y folyáshatárt, ezért ezt az értéket felhasználó feltételek „nem aktívak”. A fáradásnak ezen ciklusszám alatt nincs jelentősége. Nagyon magas ciklusszám esetén ismét konstanssá válik a költség, hiszen az $N = 10^8$ felett a fáradási határfeszültség is konstans.

Az 5.4. ábra eredményeiből már látható, hogy a mértékadó, domináns feltételek a (2.122) és a (2.123) egyenletek által maghatározott fáradási feltételek. Ezek mellett még

5.8. táblázat. Fesztáv - költség közelítőformula adatai

#	Acél minősége	Konstansok			Illeszkedési helyesség		Eltérés (RMSE)
		$\hat{C}_3$	$\hat{C}_4$	$\hat{C}_5$	R^2	korr. R^2	
1.	S235	2,29	2,728	3263	0,9986	0,9980	835,62$
2.	S355	1,93	2,783	3477	0,9984	0,9978	915,81$
3.	S460	1,66	2,851	3727	0,9985	0,9979	973,94$
4.	S690	1,00	3,072	4929	0,9988	0,9984	1188,50$



5.4. ábra. Gyártási költség változása a terhelési ciklusok számának a függvényében

5.9. táblázat. Tervezési változók értéke változó terhelési ciklus szám mellett

Terhelési ciklusok száma	Tervezési változók [mm]															
	S235				S355				S460				S690			
	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f	h	b	t_w	t_f
1,00E+05	650	390	22	10	640	384	21	11	607	364	21	16	599	359	26	15
2,00E+05	650	390	22	10	640	384	21	11	607	364	21	16	599	359	26	15
5,00E+05	650	390	22	10	640	384	21	11	607	364	21	16	599	359	26	15
8,00E+05	593	409	20	16	593	412	20	16	593	392	21	17	595	357	26	15
1,00E+06	594	416	20	18	596	417	20	18	598	419	21	17	558	389	24	20
2,00E+06	647	453	22	20	649	454	22	19	656	458	23	18	603	422	26	22
5,00E+06	751	525	25	19	753	527	25	19	764	535	26	18	691	484	30	23
8,00E+06	779	545	26	20	782	547	26	20	795	556	27	18	731	512	31	22
1,00E+07	793	555	27	20	796	557	27	20	810	567	27	19	747	523	32	22
2,00E+07	840	588	28	21	844	591	28	21	848	594	28	21	793	555	33	23
5,00E+07	909	636	31	23	913	639	31	22	918	642	31	22	859	601	36	25
8,00E+07	948	664	32	24	951	666	32	23	957	669	32	23	895	626	37	25
1,00E+08	968	677	32	24	971	680	32	24	976	683	33	23	913	638	38	26
2,00E+08	967	677	32	24	970	679	32	24	975	682	33	23	912	638	38	26
5,00E+08	966	676	32	24	971	680	32	24	974	682	33	23	912	639	38	26
8,00E+08	967	677	32	24	971	679	32	24	976	683	33	23	912	639	38	26

aktívak voltak a stabilitási kritériumok, vagyis a (2.109) - (2.111) és a (2.115) - (2.115) egyenlőtlenségek. A kis terhelési ciklusszámnál, ahol a fáradásnak kicsi a befolyásoló hatása, ezek mellé még társul a (2.125)-es lehajlási feltétel is.

A különböző anyagminőségek esetén a költséggörbék az 5.2 - 5.4. ábrákon eltolva egymással párhuzamosan vagy közel párhuzamosan futnak. Az eltolás mértéke arányos az alapanyagok kezdeti árkülönbségével. A jobb, de drágább acélminőség nem eredményez olcsóbb szerkezetet. Ennek oka, hogy a [31] és a [78] irodalmak azonos fáradási határt adnak meg a varratokra, függetlenül az alkalmazott acélminőségtől vagy folyáshatárától. Továbbá a stabilitási kritériumok esetén a folyáshatár növekedésnek káros befolyásoló

hatása van. Csökkenti a (2.112) egyenlet szerint az ε értékét, ami a megengedett t_w/h és t_f/b arányt rontja. Ez azt is jelenti, hogy szükségszerűen nő a keresztmetszeti terület.

Összességben semmi nem indokolja jelen alkalmazásban a drágább, de nagyobb folyáshatárú acélok alkalmazását. Az (5.2) és (5.3) összefüggésekkel sikerült egy olyan eszközt alkotni, amivel becsülhető egy leendő főtartó költsége. Ha a tényleges alkalmazási helyen a k_m és k_w költségtényezők el is térnek, akkor is használható egy százalékos változás becslésére.

5.3. Távvezetéktorony alsó rész optimálásának eredményei

Az optimalizálást virágbeporzási algoritmussal (FPA) és önadaptív differenciális evolúcióval (SaDE) végeztem el. A szükséges programok[7] C nyelven íródtak. Az algoritmusok paramétereit az 5.10. táblázat foglalja össze. Az optimálás során a fitneszfüggvény minden esetben folyamatosan konvergált egy értékhez mind a két algoritmus esetében. Egy példa optimalizálás konvergenciáját szemlélteti az 5.5. ábra. Jól látható, hogy mind a két esetben az $n_{IT} = 1000$ iterációs lépés sok, mivel körülbelül $n_{IT} = 200$ lépés után az optimum nem változik. Az evolúciós algoritmusok természetéből következően teljes bizonyossággal nem minősíthető a kapott eredmény, a továbbiakban globális minimumként kezelem, mert minden egyes futtatás hasonló konvergenciát mutat a vizsgált esetekben. A két algoritmus által adott eredmények között a különbségek nagyon kicsik. Az 5.11. táblázat foglalja össze ezen eredményeket a 2.8b. ábra szerinti szerkezetoptimálás során $f_y = 235\,\mathrm{MPa}$ folyáshatárú acélt alkalmazva. A kis különbségek ellenére – ami minden vizsgált esetre igaz – az SaDE adott jobb eredményt az esetek többségében. A továbbiakban csak a SaDE-vel történt optimalizálás eredményeit szemléltetem.

5.10. táblázat. Algoritmus paraméterek távvezetéktorony alsó részének optimálásához

Algoritmus	Populáció méret n_p	Maximális iteráció szám n_{IT}	Algoritmus paraméterek
SaDE	100	1000	tanulási ciklusok hossza: – $n_{lp} = 50$ – $n_{CR} = 5$ – $n_{lCR} = 25$
FPA	100	1000	valószínűségi változó: $p = 0{,}5$

Több eltérő topológiát vizsgáltam, melyek rácsosztásának a száma $n_{racs} = 1 \ldots 6$ között változott. A rudak kihasználtsága a veszélyes keresztmetszetben minden esetben 1 vagy egyhez nagyon közeli érték. Első keresztmetszetcsoportban a 2.8b. ábra jelöléseivel a mértékadó rudak 6 - 7 jelűek. A második csoportban a 16-os rúd. Végül a harmadikban pedig a 25-ös rúd. Minden topológiaváltozatnál az előzőekben említettekkel azonos pozícióban lévő rudak voltak a mértékadóak.

A 2.8b. ábra szerint a deltoid csúcspontja felez egy rácsosztáshoz tartozó oldalt. Megvizsgáltam, hogyan változik az optimalizált tömeg, ha ettől eltérünk. Az eredményeket

[7] A forráskód megtalálható a mellékelt adathordozón `./6_sz_melleklet/` könyvtárban

5.11. táblázat. A 2.8b. ábra szerinti szerkezet optimálása során kapott eredmények összehasonlítása $f_y = 235\,\text{MPa}$ folyáshatárú acélt alkalmazva

Algoritmus	Tervezési változók [kg]						Optimalizált tömeg [kg]
	D_1	D_2	D_3	t_1	t_1	t_1	
SaDE	263,78	44,76	142,13	5,28	10,10	2,84	2443,89
FPA	256,76	59,04	153,56	5,14	6,69	3,07	2496,54
Különbség (abs. érték)	7,02	14,28	11,43	0,14	3,42	0,23	52,66

szemlélteti az 5.6. ábra az h_r osztásarány változásának függvényében.

$$h_r = \frac{h_2}{h_1}\,. \tag{5.4}$$

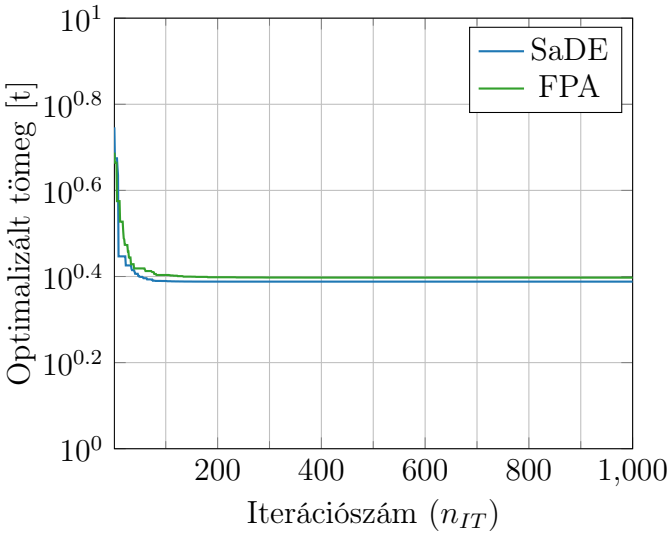
Az eredmények a vártnak megfelelően alakultak. A legkisebb tömegű szerkezetet akkor kapjuk, ha a deltoid pont felezi az adott rácsosztást (5.6a. ábra). Ha ettől eltérünk jobbra vagy balra, a tömeg másodfokú polinommal közelíthető

$$m_{opt} = 6754 \cdot (h_r - 0{,}5)^2 + 2443{,}887\,, \tag{5.5}$$

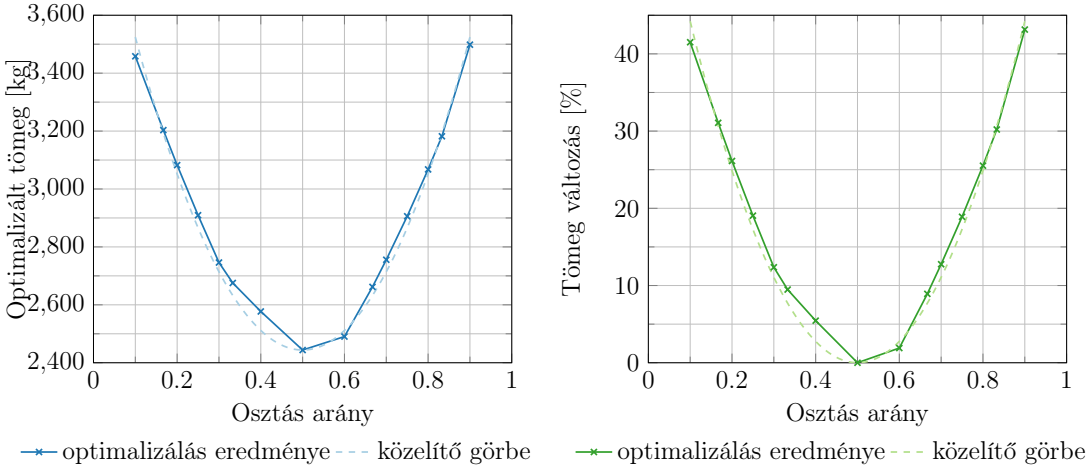
$$m_{\%} = 2{,}764 \cdot (x - 0.5)^2\,, \tag{5.6}$$

ahol m_{opt} az optimalizált tömeg kilogrammban, $m_{\%}$ a tömeg százalékos változása. Tömegnövekmény akár a 0,5 osztásarányú optimumhoz képest 40%-kal nagyobb is lehet (5.6b. ábra). Jól látható, hogy a felezőponttól eltérni nem érdemes, hacsak ezt valamilyen egyéb konstrukciós megfontolás nem indokolja.

Megvizsgáltam, hogy hogyan változik az optimált tömeg, ha különböző minőségű acélokból készül a szerkezet. Ennek az eredményét szemlélteti az 5.7. ábra. A folyáshatár növekedésével az optimumhoz tartozó rácsosztásokat adó deltoidok száma is nő. Jól látható ez a tendencia az 5.7b. ábrán, ahol a százalékos változás viszonyítási alapja az egy



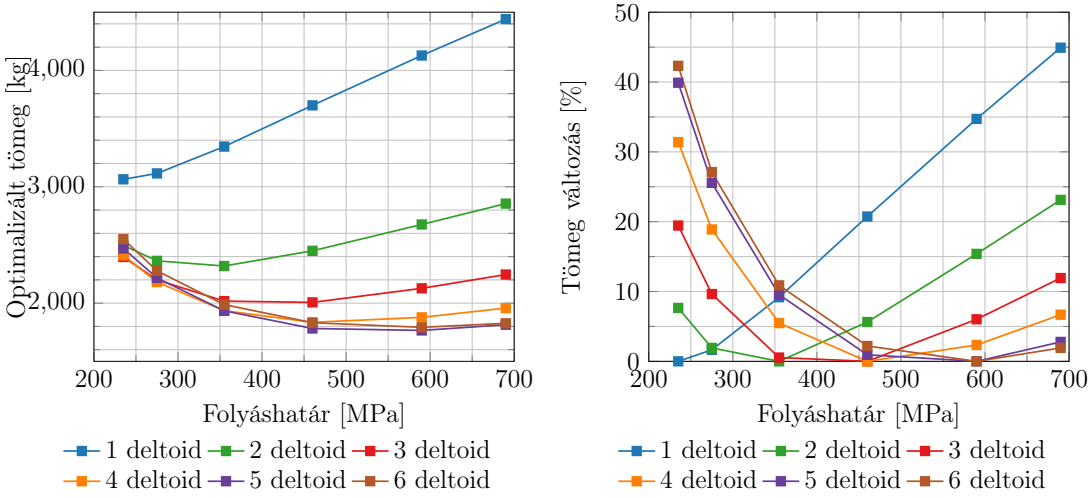
5.5. ábra. Távvezeték torony alsó részének optimálásának konvergenciája



(a) Optimált tömeg (b) Tömeg százalékos változása

5.6. ábra. Optimálás eredménye változó osztásarány mellett

deltoid számon belüli optimum. Ennek oka, hogy a folyáshatár a (2.151) egyenlet szerinti keresztmetszet kihasználtságot csökkenti. Első ránézésre kedvezően kellene hatnia a szerkezet tömegére, de befolyásolja még a (2.153) egyenlet szerinti redukált karcsúsági tényezőt is. A $\bar{\lambda}$ növekedése pedig csökkenti a χ kihajlási tényezőt. Ismét elkezd nőni a keresztmetszet-kihasználtsága. Két egymás ellen dolgozó hatást a rúdhosszak csökkenése kompenzálja, ami a deltoid szám növekedésével jár. Az f_y folyáshatár-növekedésnek a (2.154) egyenletben jelentkező káros hatása elhanyagolható. Jelen számszerűsített példában a (2.154) összefüggés lényegében csak egy felső korlátot ad a D_i átmérőknek, és alsó határt a t_i lemezvastagságoknak. Ezek inkább tekinthetőek geometria határoknak, habár az alapja a stabilitási kritériumokon nyugszik.



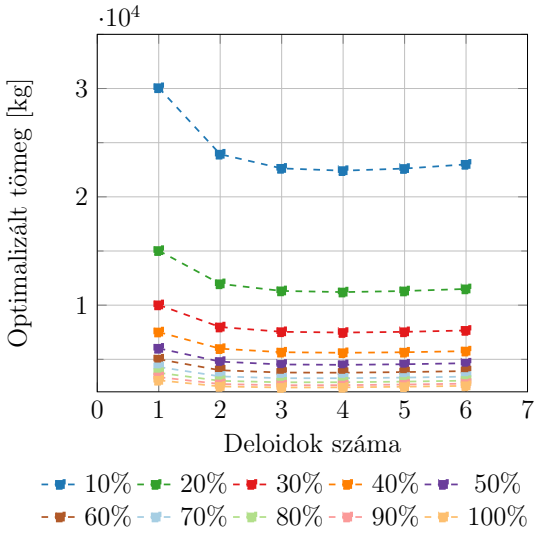
(a) Optimált tömeg (b) Tömeg százalékos változása

5.7. ábra. Optimálás eredménye változó anyagminőségekkel

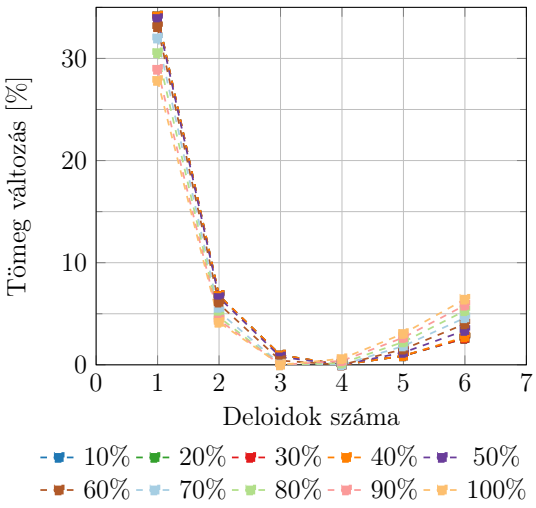
A nagyobb folyáshatárú acélból 590 MPa-ig alapvetően kisebb tömegű szerkezet készíthető (lásd a 5.7a. ábra). A nagyobb szilárdságú acél használatának létjogosultságát a helyi viszonyokhoz kötött a $\frac{k_f}{k_m}$ költségtényezők aránya dönti el. Vagyis képes-e az alapanyagárban elért megtakarítás kompenzálni a csomópontszám növekedéssel járó drasztikus

gyártási költség növekedését.

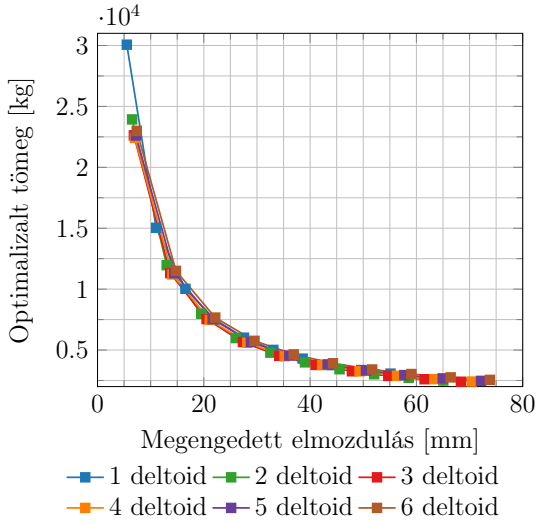
A harmadik, egyben ebben a témakörben utolsó vizsgálat tárgyát az képezi, hogy hogyan változik a szerkezet tömege, ha a 2.8b. ábra jelölései alapján a jobb felső 12-es csomópont elmozdulását korlátozzuk. A 100%-os érték jelöli a korlátozatlan elmozdulást, és a többi kisebb megengedett érték ennek az értéknek a százalékban kifejezett részaránya.



(a) Optimált tömeg



(b) Tömeg százalékos változása



(c) Tömeg változás a megengedett elmozdulás függvényében

5.8. ábra. Optimálás eredménye elmozdulás korlátozásának függvényében

Az eredményeket az 5.8. ábra szemlélteti. Függetlenül a korlát mértékétől, tömegminimum a 3 - 4 darab deltoidnál található (lásd 5.8b. ábra). A viszonyítás alapját egy adott elmozduláskorláthoz tartozó tömegminimum adja. A két deltoid szám közötti eltérés minden esetben kisebb, mint 1%. Ha az optimumot adó 3 - 4 rácsosztásnál kevesebbet alkalmazunk, akkor a tömeg gyorsan növekszik, körülbelül a 30%-os értékig. Ha pedig több kerül alkalmazásra, viszonylag lassan (lineárishoz közeli módon) nő. Az elmozduláskorlátot milliméterben megfogalmazva pedig az tapasztalható, hogy a tömeg hiperbolikusan nő, és jó közelítéssel független a rácsosztások számától ()5.8c. ábra)

$$m = 1{,}544 \cdot 10^5 x^{-1{,}01}\,, \tag{5.7}$$

ahol m a tömeg kilogrammban kifejezve, x pedig a megengedett elmozdulás milliméterben.

Összefoglalás

Nemlineáris, sokdimenziós optimalizálási és/vagy keresési feladatok megoldására jól használhatóak az evolúciós módszerek. A populáció alapú, iteratív eljárásokkal történő számítások nagy előnyét – a célfüggvényre „feketedoboz"-ként lehet tekinteni – kihasználva, a számítások könnyen automatizálhatóak, elhatárolva az algoritmust a megoldandó feladattól.

„Az evolúciós technikák és azok alkalmazása a szerkezetoptimálásban" kutatásom három fő szakaszra osztható, valamint az ezekhez kapcsolódó vizsgálatokra:

1. Mind a mai napig nagy számban publikált evolúciós algoritmusok közül kiemeltem 11 db algoritmust, és azokkal tesztfüggvények optimalizálásán keresztül szimulációt végeztem.

2. A virágbeporzási algoritmus példáján keresztül megvizsgáltam, javaslatot tettem a számítások masszívan párhuzamos feldolgozására.

3. A mérnöki gyakorlatban előforduló optimalizálási feladatot – úgymint teherautóplató kereszttartóinak, futódaru-főtartójának és rácsos tartó rúdelemeinek optimalizálása – vizsgáltam evolúciós optimalizálási módszerrel.

Az algoritmusok irodalmi összehasonlítása, teljesítményelemzése a „*no-free lunch*" elméletet is betartva nagy változékonyságot mutat, egymással nehezen összehasonlítható eredményeket publikálnak. Az általam elvégzett szimulációk során törekedtem az egységes tesztkörnyezet biztosítására. A kapott eredmények alapján jól megfigyelhető a 11 db algoritmus konvergenciája az iterációs lépések során összesen 30 tesztfüggvény esetén és 5 dimenziónagyság mellett.

A virágbeporzási algoritmus iterációs lépésen belüli számításainak mátrix és/vagy vektor műveletekre történő átalakításával, továbbá a fitneszfüggvények hierarchikus egyszerű függvényekre történő bontásával az optimalizáció futtatható SIMD architektúrán. A szekvenciális feldogozáshoz képest jelentős számítási sebességnövekedés érhető el, mely megkívánja mind az algoritmus műveleteinek, mind a fitneszfüggvényhez köthető számítások párhuzamosítását.

A teherautó-plató kereszttartóit szilárdsági, stabilitási szempontból leíró egyenletek büntetőfüggvények segítségével fitneszfüggvénybe foglalhatóak tömegminimum-keresés céljából. Az eredetileg tömeg szerint optimált RHS szelvényű kereszttartók tovább optimalizálhatóak az I-szelvények alkalmazásával és a tartók számának csökkentésével.

Szekrényszelvényű futódaru-főtartójának költség szempontjából szentjánosbogár algoritmussal történő optimálása során bemutattam, hogy a jobb minőségű, de drágább acélok alkalmazása nem javasolt. Az optimált költség a hasznos horogteher függvényében lineárisan, a fesztáv növekedésénél köbösen, és végül fáradás szempontjából a fáradási görbe szerint nő. Minden esetben a különböző acélokhoz tartózó optimalizált költségfüggvények kvázi párhuzamosak egymással.

A rácsos tartók belső erői és a méretezéshez szükséges feszültségek, meghatározhatóak végeselem-módszerrel. A végeselem-módszer elő- és utófeldolgozási műveleteinek optimalizálásával, egyszerűsítésével összekapcsoltam az önadaptív differenciális evolúció algoritmussal. A két módszer kapcsolata egy hatékony numerikus számítási eszközt kínál, melyhez saját végeselem-rendszert készítettem. Távvezetéktorony számszerűsített példáján keresztül, deltoid alakú rácsozás esetén, a tömeg szempontjából az optimális topológia – úgymint rácsosztások száma stb. – erősen függ az alkalmazott acél folyáshatárától. A nagyobb folyáshatárú acélok alkalmazása nem jelent feltétlen tömegcsökkenést.

Új tudományos eredmények összefoglalása

1. **Tézis**: Elvégeztem a 11 evolúciós algoritmus tesztelését a [67] teszt függvénykészlettel, azonos szimulációs környezetet (pl.: célfüggvény értékek kiszámításának száma, populáció mérte, stb.) alkalmazva. Az ismert optimumhoz viszonyított átlagos hibaértékek konvergenciája iterációs lépésenként, és a hibaértékek eloszlása szemléltetésre került, az algoritmusokat rangsoroltam. A rangsor és a hibaértékeloszlás együttes alkalmazása alapján, a „*no-free lunch*" elméletét betartva az algoritmusok hatékonysága és teljesítménye becsülhetővé vált a jövőbeli feladatok megoldásánál.

   Témában megjelent publikációk: ⟨1⟩, ⟨9⟩, ⟨13⟩

2. **Tézis**: Javaslatot tettem a virágbeporzási (FPA) algoritmus, és fitneszfüggvények egy csoportjának párhuzamos feldolgozására grafikus kártyákon.

   a) Javaslom a virágbeporzási algoritmus párhuzamos feldolgozásához az általa használt paraméterek – véletlen számok, input és output változók – vektorokba és mátrixokba szervezését. A mátrixként szervezett populáció elemei egymástól függetlenül párhuzamosan számíthatóak SIMD, SIMT architektúra szabályait követve.

   b) Kidolgoztam a *Sphere*, *Ackley's*, *Rastrigin* és futódaru főtartójának optimálásához szükséges fitneszfüggvények egyszerű hierarchikus függvényekre történő bontását és azok feldolgozását párhuzamos redukcióval.

   c) Bemutattam, hogy az előzőekben kidolgozott módszerekkel, mekkora dimenzióotlanított sebességnövekedés érhető el a vizsgált esetekben a párhuzamos számításokat alkalmazva.

   Témában megjelent publikációk: ⟨3⟩, ⟨11⟩, ⟨12⟩

3. **Tézis**: Kidolgoztam a teherautó plató kereszttartó optimálásához szükséges fitneszfüggvényt, és virágbeporzási algoritmust (FPA) alkalmazva kimutattam, hogy tömegminimum szempontjából az I-szelvények alkalmazása előnyösebb az eredeti RHS szelvényekhez képest, és a kereszttartók számának csökkentésével további tömegmegtakarítás érhető el.

   Kidolgoztam a szekrényszelvényű futódaru főtartójának optimalizálásához szükséges fitneszfüggvényt, melyet szentjánosbogár algoritmussal (FA) optimáltam, és kimutattam:

   a) nincs értelme nagyobb folyáshatárú, drágább acélt alkalmazni, a költségminimumhoz tartozó függvények kvázi párhuzamosak egymással,

b) a horogteher függvényében a költségfüggvény lineárisan nő,

c) a fesztáv függvényében a költségfüggvény köbös függvény szerint nő,

d) a terhelési ciklusok függvényében a költségfüggvény leköveti a fáradási görbét.

Témában megjelent publikációk: ⟨2⟩, ⟨4⟩, ⟨5⟩, ⟨10⟩, ⟨14⟩, ⟨16⟩

4. **Tézis**: Módszert javasoltam az evolúciós optimalizáláshoz szükséges fitneszfüggvény előállítására, részben végeselem-módszert használva, rácsos tartók tömegminimumra történő optimálásához az önadaptív differenciális evolúcióval. Távvezeték torony alsó részének optimálásakor deltoid alku rácsozást alkalmazva kimutattam:

   a) a deltoid alakú rácsozás kialakítása akkor a optimális, ha a rácsot alkotó rácsrudak metszéspontja az osztáson belül pont elfelezik az övrudat, ettől eltérve a tömegnövekedés akár $\approx 40\%$-os is lehet,

   b) nagyobb folyáshatárú acélok alkalmazása, a topológia megváltoztatása – úgymint rácsosztások száma – nélkül nem feltétlen eredményez kisebb tömeget,

   c) korlátozva az eredetileg legnagyobb elmozdulással rendelkező csomópont elmozdulását, a megengedett elmozdulás függvényében az optimalizált tömeg hiperbolával közelíthető.

   Témában megjelent publikációk: ⟨8⟩, ⟨15⟩, ⟨17⟩

Továbbfejlesztési irányok, lehetőségek

Az evolúciós algoritmusokkal történő optimalizálás jövőbeni fejlesztési iránya lehet – tekintettel az elvégzendő számítások mennyiségére – a párhuzamos számításokkal történő feldolgozás támogatása.

A fitneszfüggvények hierarchikus felbontásához javasolt módszer legnagyobb hátránya, hogy jelenleg kézzel történik. A megelőző kézi számítások elvégzése fáradságos, és ha nem rendszeresen előforduló, ismétlődő feladatról van szó, akár nem is éri meg elvégezni. Jövőbeni fejlesztési kutatási irányvonal lehet a szükséges kiegyensúlyozott faszerkezet előállításához algoritmus kifejlesztése, mely a matematikai kifejezésből automatikusan előállítja ezt a szükséges input vektorral együtt.

A rácsos tartók végeselem-módszerrel történő számításai során, az alkalmazott elemmodell tulajdonságaiból adódóan, az elvégzendő műveletek és megoldandó egyenletek száma viszonylag kicsi. A számuk is lassan vagy mérsékelten nő ez elemek számának növekedésével. Önmagában nézve a húzott-nyomott rúd elemekből felépített feladatmegoldását nem feltétlen igényli a párhuzamos számítással történő megoldást. Az optimálási feladatot tekintve, ahol szerkezetileg ugyanazt a csak együtthatóiban eltérő algebrai egyenletrendszert kell megoldani több ezerszer, akár tízezerszer. Jogos elvárás lehet a gyorsabb párhuzamos számítás. Populációs szinten elképzelhető, hogy az egyedenként kis méretű mátrix műveletek összefűzhetőek, utat nyitva ezzel a hatékony párhuzamos feldolgozás előtt.

Summary

Evolutionary methods are well suited for solving nonlinear, multi-dimensional optimization and / or searching problems. Population based, iterative computational procedures have a great advantage – on the target function; it can be treated as a "black box," taking advantage of being easily automated, while demarking the algorithm from the task requiring a solution.

The subject of my research is the use of evolutionary techniques in optimizing structures. My research can be divided into the following three sections and links to my investigation:

1. I highlighted 11 examples from the large volume of published evolutionary algorithms available. I performed simulations based on optimized test functions;

2. I examined the use of example pollination algorithms and recommended it for massive parallel computational processing;

3. Engineering work often requires optimization tasks – I used evolutionary methods to examine optimization opportunities on truck bed cross members, main grider of crane and truss like structures.

The literary adage of there is "no free lunch," was applied to comparing algorithms, because their outcomes are variable and difficult to compare. I strived to create uniform conditions for my simulations. The results are well observable; that the 11 chosen algorithms converge during the iterative steps with a total of 30 test functions, with a five-dimensional order of magnitude.

By converting the calculations within the iteration step of the flower pollination algorithm (FPA) to matrix and / or vector operations, and by breaking down fitness functions into hierarchical simple functions, the optimization can be run on a SIMD architecture. Compared to sequential processing, a significant increase in computational speed can be achieved, which requires parallelization of both the operations of the algorithm and the calculations associated with the fitness function.

The equations describing the cross-members of a truck's platform in terms of strength, stability and fatigue can be included in a fitness function with the help of penalty functions, for the purpose of searching for a minimum mass. Originally weight-optimized RHS crossmembers, they can be further optimized by using I-sections and reducing the number of supports.

During a cost-optimization of the main girder of a box section crane, with a firefly algorithm, it was shown that the use of steel with higher yield strength but costing more, is not recommended. The optimized cost, increases linearly as a function of payload, cubically as the span length increases, and finally according to the fatigue curve. In each case, the optimized cost functions for the different purposes are quasi-parallel to each other.

The internal forces of the truss like structures, and the stresses required for sizing, can be determined by the finite element method. By optimizing and simplifying the pre- and post-processing operations of the finite element method, I connected it with the self-adaptive differential evolution algorithm. The relationship between the two methods offers an efficient numerical calculation tool. Through the quantified example of a transmission line tower, in the case of deltoid-shaped griding, the optimal topology in terms of mass - such as the number of grids, etc. - strongly depends on the yield strength of the steel used. The use of steel with higher yield strength, does not necessarily mean a reduction in mass.

Theses

1^{st} **thesis**: I have tested 11 pcs evolutionary algorithm with test function set of [67], using the same simulation environment (e.g., number of computations of objective function values, population size, etc.). The convergence of the average error values relative to the known optimum per iteration step and the distribution of the error values were illustrated, I ranked the algorithms. Based on the ranking and distribution of errors, following the "no-free lunch" and its theory, the efficiency and performance of the algorithms in solving future tasks can be estimated.

Published pulications in this topic: ⟨1⟩, ⟨9⟩, ⟨13⟩

2^{nd} **thesis**: I proposed the parallel processing of flower pollination (FPA) algorithms and a group of fitness functions on graphics cards.

a) For the parallel processing of a flower pollination algorithm, I propose to organize the parameters – random numbers, input and output variables – into vectors and matrices. The elements of the population organized as a matrix can be calculated independently in parallel, following the rules of SIMD and / or SIMT architecture.

b) I have developed the decomposition of the fitness functions required to optimize *Sphere*, *Ackley's*, *Rastrigin* functions and main gridder of overhead crane into simple hierarchical functions and their processing by parallel reduction.

c) I have shown how the dimensionless velocity increase can be achieved with the methods developed above using the parallel calculations.

Published publications in this topic: ⟨3⟩, ⟨11⟩, ⟨12⟩

3^{rd} **thesis**: Using the fitness function required to optimize the cross-member of the truck platform, and using a flower pollination algorithm (FPA), I have shown that the use of I-sections is more advantageous than the original RHS sections in terms of minimum weight, and by reducing the number of crossmembers additional weight savings can be achieved.

I developed the fitness function required for the optimization of the main girder of the cabinet section crane, which I optimized with the firefly algorithm (FA), and showed:

a) does not make sense to use higher yield strength but more expensive steel, the cost minimum functions are quasi-parallel,

b) as a function of the hook load, the cost function increases linearly,

c) as a function of span, the cost function increases according to a cubic function,

d) as a function of load cycles, the cost function follows the fatigue curve.

Published publications in this topic: ⟨2⟩, ⟨4⟩, ⟨5⟩, ⟨10⟩, ⟨14⟩, ⟨16⟩

4^{th} **thesis**: I proposed a method to generate the fitness function required for evolutionary optimization using a finite element method to optimize truss like structures to a minimum of mass with self-adaptive differential evolution. In topic of the optimization of the lower part of the transmission line tower, I showed using deltoid shaped gridding:

a) The design of the deltoid lattice is most optimal if the point of intersection of the lattice bars forming the belt within the division is exactly half of the belt bar, otherwise the weight increase can be up to $\approx 40\%$,

b) the use of higher yield strength steels, without changing the topology, such as the number of grid division, does not necessarily result in less weight,

c) by limiting the displacement of the node with the largest original displacement, the optimized mass can be approximated by hyperbolas as a function of the allowed displacement,

Published publications in this topic: ⟨8⟩, ⟨15⟩, ⟨17⟩

Paths for further development, opportunities

Given the amount of calculations to be performed; the future development direction of optimization using evolutionary algorithms is to support processing with parallel calculations.

The biggest disadvantage of the proposed method of hierarchical division of fitness functions is, that it is currently done manually. Performing preliminary manual calculations are tedious. If they're not regularly occurring, repetitive tasks, they may not even be worth it. A direction in future development research, could be to develop an algorithm to produce the necessary balanced tree structure, that will automatically generate this from the mathematical expression, along with the required input vector.

Due to the properties of the applied element model, the number of operations to be performed and the equations to be solved are relatively small in the calculations of truss like structures, using the finite element method. Their number also increase slowly or moderately as the number of elements increase. Looking at the solution of a task made up of bar elements alone, does not necessarily require a solution with parallel computation. Regarding the optimization problem where structurally the same system of algebraic equations with different coefficients must be solved thousands of times, even ten of thousands of times, a faster parallel calculation can be a legitimate prospect.

At the population level, small individual matrix operations can be combined into a large-scale task that can be efficiently processed in parallel.

Köszönetnyilvánítás

Jelen értekezésemet ajánlom gyermekeimnek, Nagy Boglárkának és Nagy Botond Szilárdnak. Most még nem igazán értik, hogy miért olyan fontos ennek a dolgozatnak az elkészítése, miért tölt annyi időt apa számítógép előtt „pötyögve”, és miért nem lehet inkább apa hátán „lovacskázni” vagy csak birkózni. Jelen kutatásnak értékeit, elért eredményeit csak a jövőben érthetik meg, és jó esetben értékelhetik. Bízom benne, hogy erőt meríthetnek belőle álmaik, céljaik eléréséhez.

Mindenekelőtt szeretném köszönetemet kifejezni témavezetőimnek, Dr. Jármai Károlynak és Dr. Baksa Attilának, akik bevezettek a tudományos életbe, folyamatos szakmai tanácsokkal és útmutatással láttak el a kutatásaim során, építő kritikákkal segítették disszertációm elkészítését.

Köszönet illeti feleségemet, Nagyné Farkas Katót, aki folyamatos támogatást nyújtott, biztatott az elém gördülő akadályok legyőzésében. Megteremtette a tanulmányaim elvégzéséhez szükséges nyugodt családi hátteret.

Köszönöm az egri Emerson Automation FCP Kft.-nek és Emerson csoportnak a lehetőséget, a támogatást és a szükséges keretfeltételeket. Daragó Gábornak és Stańczak Łukasz Piotr-nak köszönöm, hogy elindították a tudományos pályámat.

Végül, de nem utolsó sorban köszönöm az alapos nyelvi „lektorálást” Farkasné Rácz Juliannának és Munkácsi Annának.

Kutatás témájában megjelent publikációk

Magyar nyelven megjelent publikációk:

⟨1⟩ Nagy, Sz. & Jármai, K.: Alap, hibrid és többszintű evolúciós algoritmusok, *GÉP*, 69(**2**), pp.44-51, **2018**

⟨2⟩ Nagy, Sz. & Jármai K.: Evolúciós algoritmusok és alkalmazásuk futódaru optimálásán keresztül, *Doktoranduszok fóruma 2018*, Miskolc, Hungary, **2018**

⟨3⟩ Nagy, Sz. & Jármai, K.: FPA algoritmus implementálása masszívan párhuzamos architektúrára, *GÉP*, 70(**2**), pp.16-19, **2019**

⟨4⟩ Nagy, Sz. & Jármai, K.: Futódaruhíd optimálása párhuzamos FPA algoritmussal GPU-n, *GÉP*, 71(**2**), pp.27-33, **2020**

⟨5⟩ Nagy, Sz. & Jármai, K.: Teherautó plató kereszttartójának optimálása evolúciós módszerrel, *GÉP*, 71(**3-4**), pp.86-90, 2020

⟨6⟩ Nagy, Sz. & Jármai, K. & Petrik, M. & Erdős, A. & Gafil, N. H.: Hegesztett szerkezetek tervezésének kutatása a Miskolci Egyetem, *In Gáti József. XXX. Jubileumi Nemzetközi Hegesztési Online Konferencia*, pp.1-13, **2021**

⟨7⟩ Nagy, Sz. & Jármai, K. & Petrik, M.: Miskolci Egyetem kutatási témái hegesztett szerkezetek tervezésében, *Hegesztéstechnika*, 33(**1**), pp.57-62, **2022**

⟨8⟩ Nagy, Sz & Jármai, K. & Baksa, A.: Távvezeték-torony optimálása evolúciós és VEM technikával, *GÉP*, 73(**2**), pp.17-23, **2022**

Idegen nyelven megjelent publikációk:

⟨9⟩ Nagy, Sz. & Jármai, K.: Application of the firefly algorithm for the optimization of cranes, *In Advances and Trends in Engineering Sciences and Technologies III.*, Tatranské Matliare, Slovakia, **2018**

⟨10⟩ Nagy, Sz. & Jármai, K.: Optimum design of overhead travelling crane, 3^{rd} *International Conference on Engineering Sciences and Technologies: ESAT 2018*, Tatranské Matliare, Slovakia, **2018**

⟨11⟩ Nagy, Sz. & Jármai, K.: Massively parallel flower pollination algorithm, *In MultiScience - XXXIII. microCAD International Multidisciplinary Scientific Conference*, Miskolc, Hungary, **2019**

⟨12⟩ Nagy, Sz. & Jármai, K..: Reducing computation time using GPU Based Parallelization of FPA Algorithm for Optimization, 16^{th} *Miklós Iványi International PhD & DLA Symposium*, Pécs(Online), Hungary, **2020**

⟨13⟩ Nagy, Sz. & Soltész, L.: The connection between ADT and evolutionary methods in product development, *Journal of Physics: Conference Series*, 1935(**1**), p. 012001, **2021**, doi: 10.1088/1742-6596/1935/1/012001

⟨14⟩ Nagy, Sz. & Jármai K.: GPU based parallel optimization of members of a truck floor, *Journal of Physics: Conference Series*, 1935(**1**), p. 012004, **2021**, doi: 10.1088/1742-6596/1935/1/012004

⟨15⟩ Nagy, Sz. & Jármai, K. & Baksa, A.: Evolutionary optimization of a transmission line tower with FPA algorithm, *Design of Machine and Structures*, 11(**2**), pp.36-44, **2021**

⟨16⟩ Nagy, Sz. & Jármai, K.: Using a flower pollination algorithm to optimise the cross members of a truck floor, *The* 9^{th} *International Conference on COMPUTATIONAL MECHANICS AND VIRTUAL ENGINEERING*, Brasov, Romania, **2021**

⟨17⟩ Nagy, Sz. & Jármai, K & Baksa, A.: Combining evolutionary optimization with finite element method for optimizing transmission-line tower, *IOP Conference Series: Materials Science and Engineering*, 1237(**1**), p. 012017, **2022**, doi: 10.1088/1757-899x/1237/1/012017

Irodalomjegyzék

[1] A., K. Jármai, and Gy. Kovács. Optimal design of a lightweight composite sandwich plate used for airplane containers. *Structural engineering and mechanics*, 78(5):611–622, 2021. doi: 10.12989/sem.2021.78.5.611.

[2] S. M. Abdulrahman. Using swarm intelligence for solving np-hard problems. *Academic Journal of Nawroz University*, 6:46–50, 2017. doi: 10.25007/ajnu.v6n3a78.

[3] C. Aguilar-Ibanez, F. Qian, M. R. Mahmoudi, H. Parvïn, K.-H. Pho, and B. A. Tuan. An adaptive particle swarm optimization algorithm for unconstrained optimization. *Hindawi Complexity*, 2020, 2020. doi: 10.1155/2020/2010545.

[4] S. M. Almufti. Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7:1–12, 2019.

[5] S. M. Almufti, A. Y. Zebari, and H. K. Omer. A comparative study of particle swarm optimization and genetic algorithm. *Journal of Advanced Computer Science and Technology*, 8:40–45, 2019. doi: 10.14419/jacst.v8i2.29402.

[6] A. Auger. Convergence results for the $(1, \lambda)$-sa-es using the theory of ϕ-irreducible markov chains. *Theoretical Computer Science*, 334:35–69, 2005. doi: 10.1016/j.tcs.2004.11.017.

[7] T. Back, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[8] A. Baksa. Érintkezési feladatok numerikus vizsgálata. Phd értekezés, Sályi István Gépészeti Tudományok Doktori Iskola, Miskolc, 2005.

[9] A. Baksa and I. Páczelt. Megfogófej érintkezési viszonyainak optimalizálása. In *Proc. Int. Conference on Engineering Design ICED*, pages 478–487, 1985.

[10] A. Baksa and I. Páczelt. Examination of contact optimization and wearing problems. *Journal of Computational and Applied Mechanics*, 3(1):61–84, 2002.

[11] A. Baksa, I. Páczelt, and T. Szabó. Solution of 3d contact problems using spline interpolation. *Journal of Computational and Applied Mechanics*, 9(2):125–147, 2014. doi: 10.32973/jcam.2014.007.

[12] A. Baksa, I. Páczelt, and T. Szabó. Solution of 3d contact problems using spline interpolation. *JCAM*, 9(2):125–147, 2014.

[13] P. D. Barba, F. Dughiero, M. E. Mognaschi, A. Savini, and S. Wiak. Biogeography-inspired multiobjective optimization and mems design. *IEEE Transactions on Magnetics*, 52(3):1–4, 2016. doi: 10.1109/TMAG.2015.2488982.

[14] Cs. Barcsák and K. Jármai. Optimization with an improved pso algorithm. *XXVI. MicroCAD 2012*, 2012.

[15] E. Bertóti. Primal- and dual-mixed finite element models for geometrically nonlinear shear-deformable beams – a comparative study. *Computer Assisted Methods in Engineering and Science*, 27(4):285–315, 2020. doi: 10.24423/cames.299.

[16] T. R. Biyanto, Matradji, M. N. Syamsi, H. Y. Fibrianto, N. Afdanny, A. H. Rahman, K. S. Gunawan, J. A. D. Pratama, A. Malwindasari, A. I. Abdillah, T. N. Bethiana, and Y. A. Putra. Optimization of energy efficiency and conservation in green building design using duelist, killer-whale and rain-water algorithms. 267(1): 012036, nov 2017. doi: 10.1088/1757-899x/267/1/012036.

[17] C. Blum, R. Chiong, M. Clerc, K. De Jong, Z. Michalewicz, F. Neri, and T. Weise. *Variants of Evolutionary Algorithms for Real-World Applications*. Springer Berlin Heidelberg, Berlin, 2012. ISBN 978-3-642-23424-8.

[18] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 120–127, 2007. doi: 10.1109/SIS.2007.368035.

[19] BS 2573-1:1983. Rules for the design of cranes. specification for classification, stress calculations and design criteria for structures. (standard), British Standards Institution, London, 1983.

[20] BS 8118-1:2007. Structural use of aluminium part 1: Code of practice for design. (szabvány), BSI, London, 2007.

[21] I. Burgulya. *Optimalizálás evolúciós számításokkal*. Typotex Kft., 2012. ISBN 978-963-279-680-2.

[22] J. Cheng, M. Grossman, and T. McKercher. *Professional CUDA C Programming*. John Wiley and Sons Ltd., 2014. ISBN 978-111-873-932-7.

[23] N. J. Cheung, X.-M. Ding, and H.-B. Shen. Adaptive firefly algorithm: Parameter analysis and its application. *PLOS ONE*, 9(11):1–12, 2014. doi: 10.1371/journal.pone.0112634.

[24] J.G.S. da Silva, P.C.G. da S. Vellasco, S.A.L. de Andrade, and M.I.R. de Oliveira. Structural assessment of current steel design models for transmission and telecommunication towers. *Journal of Constructional Steel Research*, 61(8):1108–1134, 2005. ISSN 0143-974X. doi: 10.1016/j.jcsr.2005.02.009. Second Brazilian special issue.

[25] S. Dan. *Evolutionary Optimization Algorithms - Biologically-Inspired and Population-Based Approaches to Computer Intelligence*. Wiley and Sons Ltd., 2013. ISBN 978-047-093-741-9.

[26] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997. doi: 10.1016/S0303-2647(97)01708-5.

[27] M. Dorigo and T. Stutzle. *Ant Colony Optimization*. MIT Press, Boston, 2004. ISBN 978-026-204-219-2.

[28] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26: 1–13, 1996.

[29] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995. doi: 10.1109/MHS.1995.494215.

[30] I. Ecsedi and A. Baksa. Deformation of a cantilever curved beam with variable cross section. *Journal of Computational and Applied Mechanics*, 16(1):23–36, 2021. doi: 10.32973/jcam.2021.002.

[31] EN 1993-1-1:2009. Eurocode 3: Design of steel structures - part 1-1: General rules and rules for buildings. (standard), European Committee Standardization, Brussels, May 2009.

[32] EN 573-1:2004. Aluminium and aluminium alloys - chemical composition and form of wrought products - part 1: Numerical designation system. (szabvány), European Committee for Standardization, Brussels, 2004.

[33] A. J. M. Ferreira. *MATLAB Codes for Finite Element Analysis*. Springer-Verlag. ISBN 978-94-007-8955-5.

[34] D. B. Fogel and L. J. Fogel. An introduction to evolutionary programming. In *Artificial Evolution*, pages 21–33, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-49948-0.

[35] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *SIMULATION*, 76(2):60–68, 2001. doi: 10.1177/003754970107600201.

[36] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley, 1989. ISBN 978-020-115-767-3.

[37] A. B. Hadj-Alouane and J. C. Bean. A genetic algorithm for the multiple-choice integer program. *Operations Research*, 42:92–101, 1997. doi: 10.1287/opre.45.1.92.

[38] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003. doi: 10.1162/106365603321828970.

[39] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad. Emperor penguins colony: a new metaheuristic algorithm for optimization. *Evolutionary Intelligence*, 12:211–226, 2019. doi: 10.1007/s12065-019-00212-x.

[40] A. Hobbacher. *Recommendations for Fatigue Design of Welded Joints and Components IIW-1823-07/XIII-2151r4-07/XV-1254r4-07.* International Institute of Welding, Paris, 2008.

[41] A. Homaifar, C. X. Qi, and S. H. Lai. Constrained optimization via genetic algorithms. *Simulation*, 62:242–254, 1994.

[42] L. Horrigue, R. Ghodhbane, T. Saidani, and M. Atri. GPU acceleration of image processing algorithm based on matlab cuda. *IJCSNS International Journal of Computer Science and Network Security*, 18:91 – 99, 06 2018.

[43] D. Huri and T. Mankovits. Comparison of the material models in rubber finite element analysis. *IOP Conference Series: Materials Science and Engineering*, 393: 012018, 2018. doi: 10.1088/1757-899x/393/1/012018.

[44] P. Horváth I. Timár, T. Borbély. Optimierung von profilierten sandwichbalken. *Stahlbau*, 72(2):109–113, 2003. doi: 10.1002/stab.200300330.

[45] A. Iványi. *Párhuzamos algoritmusok.* ELTE IT, 2010. ISBN 9634635903.

[46] J. Jalkanen. 2.3 - multicriteria tubular truss optimization. In Károly Jármai and József Farkas, editors, *Design, Fabrication and Economy of Welded Structures*, pages 71–78. Woodhead Publishing, 2008. ISBN 978-1-904275-28-2. doi: 10.1533/9781782420484.2.71.

[47] K. Jármai and J. Farkas. Cost calculation and optimisation of welded steel structures. *Journal of Constructional Steel Research*, 50(2):115–135, 1999. ISSN 0143-974X. doi: 10.1016/S0143-974X(98)00241-7.

[48] K. Jármai and J. Farkas. Optimum design and cost calculation of a simple frame with welded or bolted corner joints. *Welding in the World*, 48:1878–6669, 2004. doi: 10.1007/BF03266413.

[49] K. Jármai and J. Farkas. *Fémszerkezetek innovatív tervezése.* Gazdász Elasztik Kiadó és Nyomda, 2015. ISBN 978-963-358-064-6.

[50] K. Jármai and J. Farkas. Truck floor design for minimum mass and cost using different materials. In *Vehicle and Automotive Engineering*, pages 13–25, Cham, 2017. Springer International Publishing.

[51] S. Jason and K. Edward. *CUDA by Example - An Introduction to general-purpose GPU Programming.* Addison-Wesley, 2010. ISBN 978-013-138-768-3.

[52] J. A. Joines and C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, 2:579–584, 1994.

[53] K. Jármai and J. Farkas. *Desing and Optimization of Metal Structures.* Horwood Publishing, 2008. ISBN 978-1-904275-29-9.

[54] K. Jármai and J. Farkas. *Optimum Design of Steel Structures.* Springer-Verlag, 2013. ISBN 978-3-642-36867-7. doi: 10.1007/978-3-642-36868-4.

[55] K. Jármai, Cs. Barcsák, and G. Z. Marcsák. A box-girder design using metaheuristic algorithms and mathematical test functions for comparison. *Applied Mechanics*, 2 (4):891–910, 2021. ISSN 2673-3161. doi: 10.3390/applmech2040052.

[56] D. K. and B. Akay. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8:687–697, 2007. doi: 10.1016/j.asoc.2007.05.007.

[57] Beelich K. H. and Pahl G. Kostenwachstumsgesetze nach Ähnlichkeitsbeziehungen für schweißverbindungen. *VDI-Berichte*, 457:129–141, 1992.

[58] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214:108–132, 2009. doi: 10.1016/j.amc.2009.03.090.

[59] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39:359–471, 2007. doi: 10.1007/s10898-007-9149-x.

[60] S. Kazarlis and V. Petridis. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *Parallel Problem Solving from Nature — PPSN V*, pages 211–220, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[61] V. Kindratenko. *Numerical Computations with GPUs*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-37994-4.

[62] U. Klansek and S. Kravanja. Cost estimation, optimization and competitiveness of different composite floor systems—part 2: Optimization based competitiveness between the composite i beams, channel-section and hollow-section trusses. *Journal of Constructional Steel Research*, 62(5):449–462, 2006. ISSN 0143-974X. doi: 10.1016/j.jcsr.2005.08.006.

[63] L. Kota and K. Jármai. Application of multilevel optimization algorithms. In *Advances in Structural and Multidisciplinary Optimization*, pages 710–715, Cham, 2018. Springer International Publishing. ISBN 978-3-319-67988-4.

[64] L. Kota and K. Jármai. Application of a multilevel firefly algorithm on a large variable number logistic problem. *Advanced Logistic Systems - Theory and Practice*, 13(2):21–28, 2021. doi: 10.32971/als.2020.002.

[65] T. Kulcsár and I. Tímár. Mathematical optimization and engineering applications. *Mathematical Modeling and Computing*, 3(1):59–78, 2016. doi: 10.23939/mmc2016.01.059.

[66] X.-L. Li, J.-S. Wang, and X. Yang. Invasive weed optimization algorithm based on differential evolution operators to solve bin packing problem. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 4141–4145, 2020. doi: 10.1109/CCDC49329.2020.9164817.

[67] J.-C. Liang, B. Qu, and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, Beijing, 2014. IEEE.

[68] T. Mankovits. Basic principles of shape optimization of elastomers. *International Review of Applied Sciences and Engineering*, 2:75 – 78, 2011.

[69] T. Mankovits and T. Szabó. Finite element analysis of rubber bumper used in air-springs. *Procedia Engineering*, 48:388–395, 2012. ISSN 1877-7058. doi: 10.1016/j.proeng.2012.09.530. Modelling of Mechanical and Mechatronics Systems.

[70] T. Mankovits, T. Szabó, I. Kocsis, and I. Páczelt. Optimization of the shape of axisymmetric rubber bumpers. *Strojniški vestnik - Journal of Mechanical Engineering*, 60(1):61–71, 2014. ISSN 0039-2480. doi: 10.5545/sv-jme.2013.1315.

[71] R. N. Mantegna. Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes. *Phys. Rev. E*, 49:4677–4683, May 1994. doi: 10.1103/PhysRevE.49.4677.

[72] P. J. Martín, L. F. Ayuso, R. Torres, and A. Gavilanes. Algorithmic strategies for optimizing the parallel reduction primitive in cuda. In *2012 International Conference on High Performance Computing Simulation (HPCS)*, pages 511–519, 2012. doi: 10.1109/HPCSim.2012.6266966.

[73] A. R. Mehrabian and C. Lucas. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, 1(4):355–366, 2006. ISSN 1574-9541. doi: 10.1016/j.ecoinf.2006.07.003.

[74] Z. Michalewicz, D. Dasgupta, R. G. Le Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problem. *Computers and Industrial Engineering*, 30:851–870, 1996. doi: 10.1016/0360-8352(96)00037-X.

[75] S. Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, 27:1053–1073, 2016. doi: 10.1007/s00521-015-1920-1.

[76] M. Misaghi and M. Yaghoobi. Improved invasive weed optimization algorithm (iwo) based on chaos theory for optimal design of pid controller. *Journal of Computational Design and Engineering*, 6(3):284–295, 2019. doi: 10.1016/j.jcde.2019.01.001.

[77] MSZ 151-1:2000. Erősáramú szabadvezetékek. 1 kv-nál nagyobb névleges feszültségű szabadvezetékek létesítési előírásai. (szabvány), Magyar Szabványügyi Testület, Budapest, June 2000.

[78] MSZ EN 13001-3-1:2012+A1:2013. Daruk. Általános kialakítás. 3-1. rész: Acélszerkezetek határállapotai és megfelelőségigazolása. (szabvány), Magyar Szabványügyi Testület, Budapest, 2013.

[79] M. Neshat, B. Alexander, N. Y. Sergiienko, and M. Wagner. Optimisation of large wave farms using a multi-strategy evolutionary framework. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 1150–1158, 2020. ISBN 978-145-037-128-5. doi: 10.1145/3377930.3390235.

[80] F. Orbán, J. Farkas, and K. Jármai. Optimum design of a transmission line tower: Welded tubular truss structure. In *6th European Conference on Steel and Composite Structures, Eurosteel 2011*, ECCS, pages 2325–2330, 2011. ISBN 978-92-9147-103-4.

[81] H. H. Ott and V. Hubka. Vorausberechnung der herstellkosten von schweisskonstruktionen (fabrication cost calculation of welded structures). In *Proc. Int. Conference on Engineering Design ICED*, pages 478–487, 1985.

[82] I. Páczelt. Some optimization problems of contact bodies within the linear theory of elasticity. In S. NEMAT-NASSER, editor, *Variational Methods in the Mechanics of Solids*, pages 349–356. Pergamon, 1980. ISBN 978-0-08-024728-1. doi: 10.1016/B978-0-08-024728-1.50061-6.

[83] I. Páczelt. *Végeselem-módszer a mérnöki gyakorlatban.* Miskolci Egyetemi Kiadó, 1999. ISBN 963-661-312-5.

[84] I. Páczelt and A. Baksa. Examination of contact optimization and wearing problems. *Journal of Computational and Applied Mechanics*, 3(1):61–84, 2002.

[85] I. Páczelt, Z. Mroz, and A. Baksa. Analysis of steady wear processes for periodic sliding. *JCAM*, 10(2):231–268, 2015.

[86] I. Páczelt, A. Baksa, and Z. Mróz. Analysis of steady wear state of the drum brake. *Open Access Library Journal*, 7:1–16, 2020. doi: 10.4236/oalib.1106432.

[87] I. Páczelt, A. Baksa, and T. Szabó. Formulation of p-extension finite elements for solution of the normal contact problems. *Journal of Computational and Applied Mechanics*, 15(2):135–172, 2020. doi: 10.32973/jcam.2020.009.

[88] I. Páczelt, A. Baksa, and T. Szabó. Formulation of p-extension finite elements for solution of the normal contact problems. *JCAM*, 15(2):135–172, 2020.

[89] Q.-K. Pana, M. F. Tasgetirenb, P. N. Suganthanc, and T. J. Chuad. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, 181:2455–2468, 2011. doi: 10.1016/j.ins.2009.12.025.

[90] M. Petrik and K. Jármai. Optimization and comparison of different standards for compressed welded box columns. *Pollack Periodica*, 15(1):3–14, 2020. doi: 10.1556/606.2020.15.1.1.

[91] M. Petrik, A. Erdos, K. Jármai, and G. L. Szepesi. Optimum design of an air tank for fatigue and fire load. *Acta Polytechnica Hungarica*, 18:163–177, 2021.

[92] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm technical note. *Manufacturing Engineering Centre, Cardiff University, UK*, page 1–57, 09 2005.

[93] D. T. Pham and M. Castellani. The bees algorithm: Modelling foraging behaviour to solve continuous optimization problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223: 2919–2938, 2009. doi: 10.1243/09544062JMES1494.

[94] D. T. Pham and M. Castellani. A comparative study of the bees algorithm as a tool for function optimisation. *Cogent Engineering*, 2:1–28, 2015. doi: 10.1080/23311916.2015.1091540.

[95] N. Prasad Rao, G.M. Samuel Knight, S.J. Mohan, and N. Lakshmanan. Studies on failure of transmission line towers in testing. *Engineering Structures*, 35:55–70, 2012. ISSN 0141-0296. doi: 10.1016/j.engstruct.2011.10.017.

[96] A.K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791, 2005. doi: 10.1109/CEC.2005.1554904.

[97] G. V. Rao. Optimum designs for transmission line towers. *Computers & Structures*, 57(1):81–92, 1995. ISSN 0045-7949. doi: 10.1016/0045-7949(94)00597-V.

[98] N. P. Rao, G. M. S. Knight, N. Lakshmanan, and N. R. Iyer. Investigation of transmission line tower failures. *Engineering Failure Analysis*, 17(5):1127–1141, 2010. ISSN 1350-6307. doi: https://doi.org/10.1016/j.engfailanal.2010.01.008.

[99] M.P. Saka, O. Hasançebi, and Z.W. Geem. Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm and Evolutionary Computation*, 28:88–97, 2016. doi: 10.1016/j.swevo.2016.01.005.

[100] B. Schmidt, J. Gonzalez-Dominguez, C. Hundt, and M. Schlarb. *Parallel Programming*. Morgan Kaufmann, 2018. ISBN 978-0-12-849890-3.

[101] X. H. Shi, Y.H. Lu, C. G. Zhou, H. P. Lee, W. Z. Lin, and Y. C. Liang. Hybrid evolutionary algorithms based on pso and ga. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 4, pages 2393–2399 Vol.4, 2003. doi: 10.1109/CEC.2003.1299387.

[102] D. Simon. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6):702–713, 2008. doi: 10.1109/TEVC.2008.919004.

[103] D. Simon. A dynamic system model of biogeography-based optimization. *Applied Soft Computing*, 11(8):5652–5661, 2011. doi: 10.1016/j.asoc.2011.03.028.

[104] I. M. Smith and L. Margetts. *Programming the Finite Element Method, 4th edition*. John Wiley and Sons Ltd. ISBN 0-470-84969-3.

[105] K. Sorensen, M. Sevaux, and F. Glover. A history of metaheuristics. *arXiv preprint arXiv:1704.00853*, 2017.

[106] W. Steingartner and I. Yar-Muhamedov. Learning software for handling the mathematical expressions. *Journal of Applied Mathematics and Computational Mechanics*, 17(2):77–91, 2018. doi: 10.17512/jamcm.2018.2.07.

[107] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization volume*, 11:341–359, 1997. doi: 10.1023/A:1008202821328.

[108] F. J. Szabó. Optimumkereső algoritmusok iterációtörténetének vizsgálata. *GÉP*, 69:82 – 85, 2018.

[109] F. J. Szabó. Iteration history analysis of evolutionary type optimization algorithms. In *Proceedings of the 13th World Congress of Structural and Multidisciplinary Optimization (WCSMO 13)*, Beijing, 2019. Springer.

[110] K. Taniwaki and S. Ohkubo. Optimal synthesis method for transmission tower truss structures subjected to static and seismic loads. *Structural and Multidisciplinary Optimization*, 26(6):441–454, 2004. doi: 10.1007/s00158-003-0367-7.

[111] S. L. Tilahun, J. Medard, and T. Ngnotchouye. Firefly algorithm for discrete optimization problems: A survey. *KSCE Journal of Civil Engineering*, 21:535–545, 2017. doi: 10.1007/s12205-017-1501-1.

[112] W. M. K. Tizani, K. O. Yusuf, G. Davies, and N.J. Smith. A knowledge based system to support joint fabrication decision making at the design stage – case studies for chs trusses. In *Tubular Structures VII: Proceedings of the seventh international symposium*.

[113] Z. Virág and K. Jármai. Optimum design of stiffened plates for static and dynamic loadings using different ribs. *Structural engineering and mechanics*, 6(2):255–266, 2020. doi: 10.12989/sem.2020.74.2.255.

[114] Z. Virág and S. Szirbik. Finite element analysis of an optimized hybrid stiffened plate. In 9^{th} *edition of the International Multidisciplinary Symposium UNIVERSITARIA SIMPRO 202*.

[115] Z. Virág and S. Szirbik. Modal analysis of optimized trapezoidal stiffened plates under lateral pressure and uniaxial compression. *Applied Mechanics*, 2(4):681–693, 2021. ISSN 2673-3161. doi: 10.3390/applmech2040039.

[116] Z. Virág and S. Szirbik. Hibrid bordázott lemezek rezonancia vizsgálata végeselemmódszerrel. In *Multidiszciplináris Tudományok*, volume 11, pages 32–37, 2021. doi: 10.35925/j.multi.2021.2.5.

[117] T. Wang and Q. Kemao. *GPU Acceleration for Optical Measurement*. SPIE Press. ISBN 978-151-061-734-6.

[118] J. Wardenier, Y. Kurobane, J. A. Packer, A. van der Vegte, and X.-L. Zhao. *Design guide for circular hollow section (CHS) joints under predominantly static loading*. CIDECT, 2008. ISBN 978-3-938817-03-2.

[119] X.-S. Yang. Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications*, volume 5792, pages 169–178, Berlin, 2009. Springer.

[120] X.-S. Yang. Flower pollination algorithm for global optimization. In Jérôme Durand-Lose and Nataša Jonoska, editors, *Unconventional Computation and Natural Computation*, pages 240–249, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-32894-7.

[121] X.-S. Yang. *Cuckoo Search and Firefly Algorithm - Theory and Applications*. Springer International Publishing, 2014. ISBN 978-331-902-140-9.

[122] X.-S. Yang. Chapter 9 - cuckoo search. In X.-S. Yang, editor, *Nature-Inspired Optimization Algorithms*, pages 129–139. Elsevier, Oxford, 2014. ISBN 978-0-12-416743-8.

[123] X.-S. Yang, M. Karamanoglu, and X. He. Flower pollination algorithm: A novel approach for multiobjective optimization. *Engineering Optimization*, 46(9):1222–1237, 2014. doi: 10.1080/0305215X.2013.832237.

[124] Z. Yang, K. Tang, and X. Yao. Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1110–1116, 2008. doi: 10.1109/CEC.2008.4630935.

[125] Z. Yang, X. Yao, and J. He. Making a difference to differential evolution. In *Advances in Metaheuristics for Hard Optimization*, 2008.

[126] M. Yazdani and F. Jolai. Lion optimization algorithm (loa): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, 3:24–36, 2015.

[127] O. Yeniay. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10:45–56, 2005.