Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

ADAPTÍV TŰZIJÁTÉK ALGORITMUS ALKALMAZÁSA KÉTDIMENZIÓS INVERZ HŐKÖZLÉSI PROBLÉMA MEGOLDÁSÁRA

Szerző(k): **Szabó-Gali Ákos**
mérnökinformatikus BSc. szak, III. évf.

Konzulens(ek): **Dr. habil. Felde Imre Gábor**
egyetemi docens
Dr. habil. Szénási Sándor
egyetemi docens

Budapest, 2020.

Tartalomjegyzék

1. Bevezetés és célkitűzések

Számos gyártási vagy üzemeltetési folyamatot kísérő hőátadási jelenség befolyásolja magának a folyamatnak a kimenetelét. Jellegzetesen ilyen folyamatnak tekintendő a fémek hőkezelése, melynek célja és eredménye – mint már az elnevezése is mutatja – a hőmérséklettől, a hőmérséklet változásától erősen függ. Az acélok bemerítéses edzésekor (immersion quenching) a magas hőmérsékletű (800-900°C) alkatrészeket hűtőfolyadék(ok)ba mártják és így a munkadarabok a gyors lehűlés eredményeként nyerik el az elvárt tulajdonságaikat, jellemzően a különböző mechanikai igénybevételekkel szembeni megnövelt ellenállóképességüket. A bemerítés közben a hőelvonás sebessége és mértéke számos paramétertől függ (folyadék típusa, hőmérséklete, áramlási viszonyai, az alkatrész hőmérséklete, geometriája, felületi minősége, stb). A hőkezeléssel elérni kívánt előnyös tulajdonságok abban az esetben alakíthatók ki biztonságosan, ha ismert a hűtőfolyadék hőelvonási karakterisztikája.

Az előbb röviden felvázolt példa arra mutat rá, hogy a hőátadás számszerű jellemzésére napjainkban is szükség van.

A dolgozat célja egy olyan numerikus eljárás kifejlesztése, mely az időben változó hőátadás számszerű jellemzésére hivatott Hőátadási Együtthatók becslésére alkalmas. A kidolgozott predikciós eljárás központi eleme az Adaptív Tűzijáték algoritmus (AFWA, Adaptive Fireworks Algorithm), melyet az inverz hőátadási probléma megoldásához használtam fel.

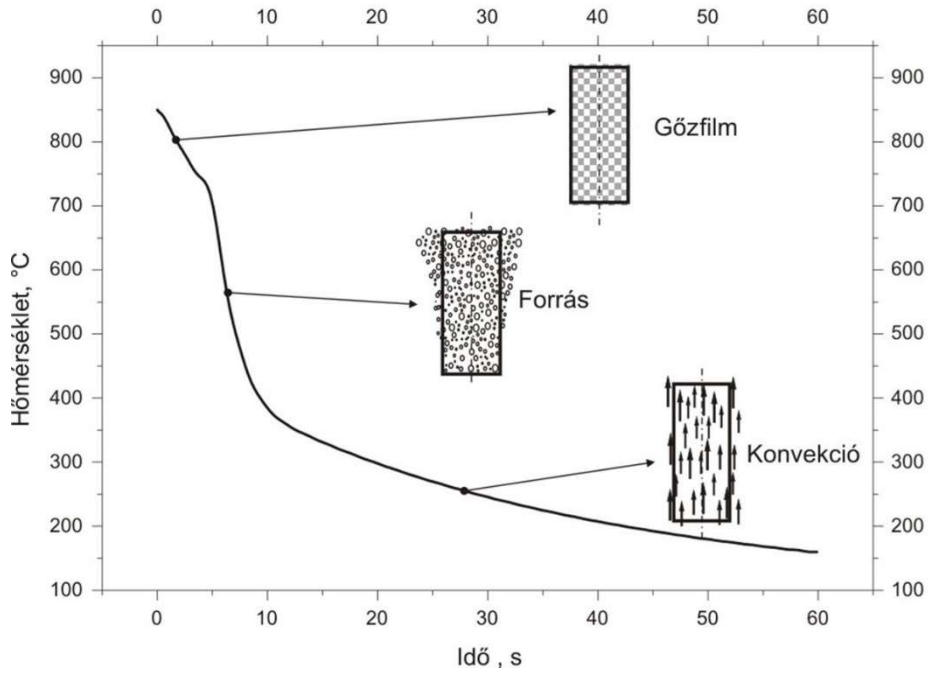
A dolgozatom felépítése az alábbi: a bevezetést követő második fejezetben a fémek bemerítéses edzése közben lejátszódó hőátadási jelenséget mutatom be. A Hőátadási Együttható becsléséhez alkalmazott módszertanal a harmadik fejezetben foglalkozom részletesen. , A hőátadás modellezéséhez szükséges kétdimenziós matematikai modellt ismertetem, melynek alkalmazásával egy tengelyszimmetrikus (hengeres) test hőmérsékletváltozása számítható ki, valamint az inverz hőátadási feladatok megoldási lehetőségéről és ehhez kapcsolódóan a Tűzijáték Algoritmusról, illetve változatairól ejtek szót. A negyedik fejezetben tárgyalom az általam kidolgozott párhuzamosított és implementált numerikus módszer részleteit, illetve ismertetem a GPU implementáció előnyeit. Az ötödik fejezetben az AFWA alkalmazásával működő numerikus eljárás vizsgálatának eredményeit elemzem, amely során Genetikus Algoritmus alkalmazásával hasonlítom össze. Majd végül a módszer felhasználhatóságát egy

organikus olajban történt bemerítéses edzés során kialakuló hőátadás becslésén keresztül demonstrálom. A munkám eredményeit a hatodik fejezetben foglalom össze.

2. Bemerítéses edzés során végbemenő hőátadási jelenség

A bemerítéses hűtéses edzés során létrejövő hőátadási folyamatokat a hűtési mechanizmusuk tekintetében három szakaszra lehet bontani (amennyiben a munkadarab felületi hőmérséklete számottevően meghaladja a hűtőközeg hőmérsékletét). Ezek a szakaszok a gőzképződési vagy Gőzfilm (vapour blanket), Forrás (nucleate boiling), illetve a Konvekciós (convection) szakaszok. [1]

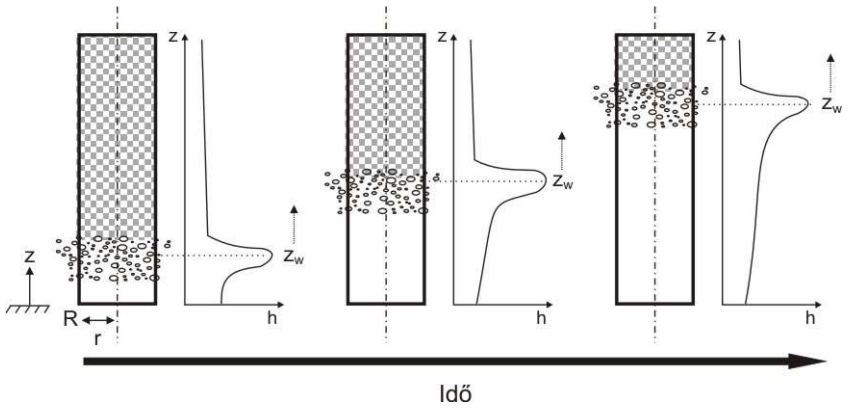
Nem sokkal a munkadarab hűtőközegbe való helyezése után, gőzfázisú film réteg keletkezik a munkadarab felületén, amely magasabb „hőellenálló" képessége miatt egyfajta hőszigetelő rétegként van jelen, vagyis lassítja a hőátadást. A „Gőzfilm" hőátadási szakaszban a hő sugárzás vagy a konvekció útján távozik a filmrétegen keresztül. A gőzfilm elvékonyodása után, a Leidenfrost hőmérsékleten a gőz hártya felszakadozik és elkezdődik a közeg forrása. [2] A „Forrás" szakaszban a munkadarab felszínének hőmérséklete a Leidenfrost hőmérséklet alá süllyed és az edzőközeg forrása következtében a gőzfilm teljesen eltűnik. A forrási jelenség a hűteni kívánt tárgy felületének a forráspont alá való hűlésével megszűnik és elkezdődik a „Konvekciós" hőátadási szakasz.



2.1 ábra A folyadék hőelvonási szakaszai hengeres test palástja mentén

A hűtőközegbe helyezett test felületén a hűtési folyamat „dinamikájára” jellemző szakaszok (gőzfilm képződési, forrási, konvekciós szakaszok) időben egymás utáni sorrendben jelentkeznek. A szakaszok kezdetének időpontja és időtartama jelentősen eltérő lehet a hűtési körülményektől függően, még az egyazon munkadarab felületi pontjain is. Egy hengeres geometriájú munkadarabon ez a jelenség jól szemléltethető. A 2.2 ábrán egy rúd alakú próbatest felszíni pontjait vizsgáljuk a henger alsó körlapjától való távolság függvényében, ezt a távolságot a „z” hely-koordináta reprezentálja (a henger alján z = 0). A felhevített munkadarab edzőközegbe merítésekor a próbatest teljes felszínén gőzfilm alakul ki, amely a hűtés mechanizmusának tekintetében az első szakasszal feleltethető meg. Az első szakasz befejeztével elkezdődik a forrási szakasz, igen gyakran [1] a hengerpalást alján (a z = 0, a r = R koordinátájú körvonal mentén. A hűlés előrehaladtával a gőzfilm folyamatosan szakadozik fel a palást felületén. A gőz felszakadás helye a „Nedvesítési front” elnevezéssel szerepel a szakirodalomban [3], mely arra utal, hogy a film eltűnésével a folyadék ettől fogva érintkezik a munkadarab felszínével. A 2.2 ábrán a nedvesítési front mozgása és a henger palástjára jellemző Hőátadási Együttható értékének eloszlása követhető nyomon három, egymást követő pillanatban (balról jobbra haladva az idő elteltével). Az rúd felületén egyszerre van jelen mind a három hőátadási szakasz. A konvekciós szakasz a henger alsó részén, ehhez képest feljebb a forrási szakasz, amely éppen felfelé mozog és szakítja fel a kialakult Gőzfilmet (mely a henger felső részét fedi). A hőátadási szakaszok pillanatnyi elhelyezkedése melletti Hőátadási

Együttható értékék arra utalnak, hogy a legnagyobb mértékű hőátadás a Forrásnál, ennél lényegesen kisebb a Konvekciós, míg a legkisebb a Gőzfilmnél adódik. Ez az ábra összességében azt szemlélteti, hogy a bemerítéses edzésnél a hőátadás időben és térben (hely-koordinátához rendelhetően) változik.



2.2 ábra A nedvesítési front (z_w) helyzete és mozgása

A nedvesítési front mozgása által meghatározott hőátadás következtében a henger alkotója mentén (a felülettől mért távolság függvényében) inhomogén hőmérsékleteloszlás jön létre, amely inhomogén szerkezeti és mechanikai tulajdonságok kialakulásához vezet. Az előbbiekben részletezett példából és gondolatmenetből következtethető, hogy az edzés folyamán végbemenő felületi hőátadás olyan bonyolult dinamikus folyamat, amelynek a komplex szemléletű, matematikai leírása jelenleg is a hőközlésre fókuszáló kutatások gyakori célkitűzése. Ezzel összhangban a dolgozatom célkitűzése az idő és hely függvényében változó (temporo-spatiális) hőátadási folyamat kvantitatív értékelésére alkalmas számítási módszer kidolgozása.

3. A temporo-spatiális Hőátadási Együttható becslésének módszertana

Az alábbiakban az idő és a hely függvényében változó Hőátadási Együttható számítására kidolgozott módszert és annak fő elemeit mutatom be. Részletesen foglalkozom egy tengelyszimmetrikus test kétdimenziós hőátadási modelljével és annak véges differencia megoldásával. Ezután a termikus peremfeltételben szereplő Hőátadási Együttható függvények becslésére kidolgozott, az inverz hőátadási probléma (IHCP, Inverse Heat Conduction

Problem) megoldására javasolt heurisztikáról ejtek szót. Végül az IHCP megoldására kidolgozott számítási eljárás központi elemét, az Adaptív Tűzijáték Algoritmust mutatom be.

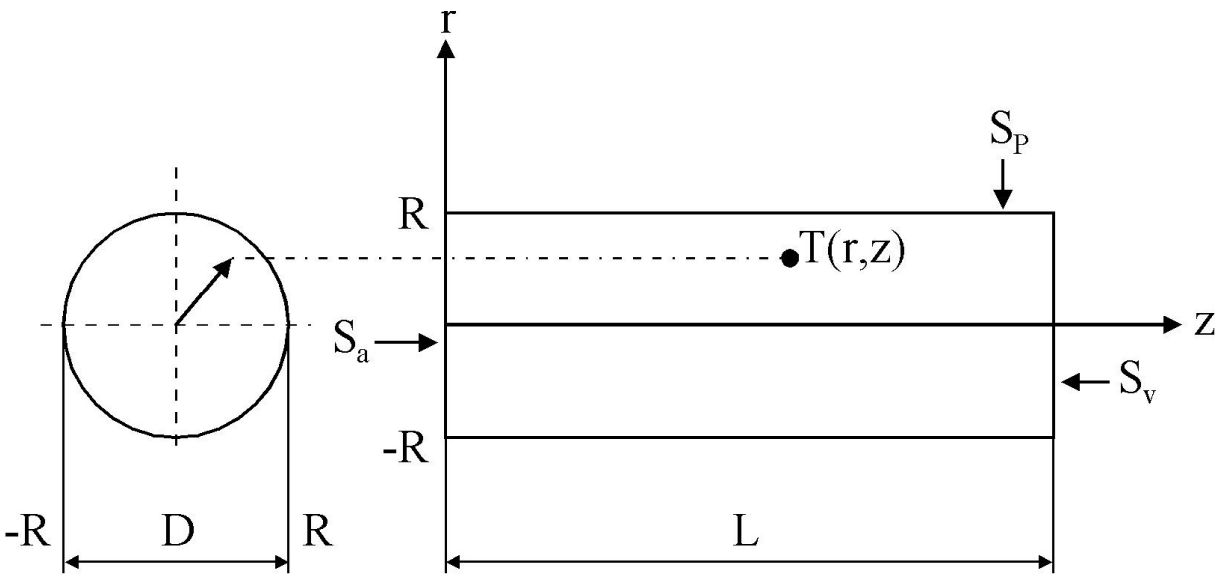
3.1. Tengelyszimmetrikus, hengeres test hőátadási modellje

A hőátadási együttható becslésére kidolgozott numerikus eljárást egy tengelyszimmetrikus hengertest lehűlésén teszteltem. Ebben az alfejezetben a 2D hőátadást feltételező tengelyszimmetrikus henger hőátadási matematikai modellt ismertetem röviden.

A hőközlési folyamatot véges L hosszúságú D=2R átmérőjű hengerre centrál-szimmetrikus hőátadási viszonyokat feltételezve az alábbi differenciálegyenlet írja le:

$$\frac{\partial}{\partial r}\left(\lambda\frac{\partial T}{\partial r}\right)+\frac{\lambda}{r}\frac{\partial T}{\partial r}+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+q_v=\rho C_p\frac{\partial T}{\partial t} \qquad (2.1)$$

ahol t az idő, r a polár koordináta, z a helykoordináta "z" tengely mentén, T(r,z,t) a hőmérséklet, q_v a látens hőmennyiség (ennek értéke a kutatási munkánk során a teljes hőátadási folyamat közben nulla), ρ a sűrűség, C_p a fajhő, λ a hővezetési tényező (2.3 ábra).



2.3 ábra A hengeres test modellje

A kezdeti feltétel megadására a

$$T(r, z, 0) = T_a \qquad (2.2)$$

egyenlet szolgál, ahol T_a a hűlést megelőző (kezdő) hőmérséklet. A peremfeltételeket - a 2.1 ábra jelölési rendszerével összhangban - az alábbi egyenletek reprezentálják.

A henger tengelyére a hőátadás körszimmetrikus jellegéből adódóan

$$\lambda \frac{\partial T}{\partial r}\Big|_{r=0} = 0 \qquad (2.3.1)$$

összefüggés teljesül. A hengert határoló három részfelületen (ahol S_p a henger palástja, S_a illetve S_v pedig a henger alap- valamint véglapja) a hőátadást a

$$\lambda \frac{\partial T}{\partial z}\Big|_{z=0} = h_v\left[T_q - T(r, z = 0, t)\right], \qquad (2.3.2)$$

$$\lambda \frac{\partial T}{\partial z}\Big|_{z=L} = h_a\left[T_q - T(r, z = L, t)\right], \qquad (2.3.3)$$

$$\lambda \frac{\partial T}{\partial r}\Big|_{r=R} = h_p\left[T_q - T(r = R, z, t)\right] \qquad (2.3.4)$$

egyenletek írják le. A fenti egyenletekben T_q a hűtőközeg környezeti hőmérséklet, míg

$$h_v = h_v(t, r) \qquad (z = 0) \qquad (2.3.5)$$

$$h_a = h_a(t, r) \qquad (z = L) \qquad (2.3.6)$$

$$h_p = h_p(t, z) \qquad (r = R) \qquad (2.3.7)$$

a felületi hőmérséklettől és a felületi hely-koordinátától is függő, az S_p, S_a valamint S_v felületelemekre vonatkozó hőátadási együtthatók.

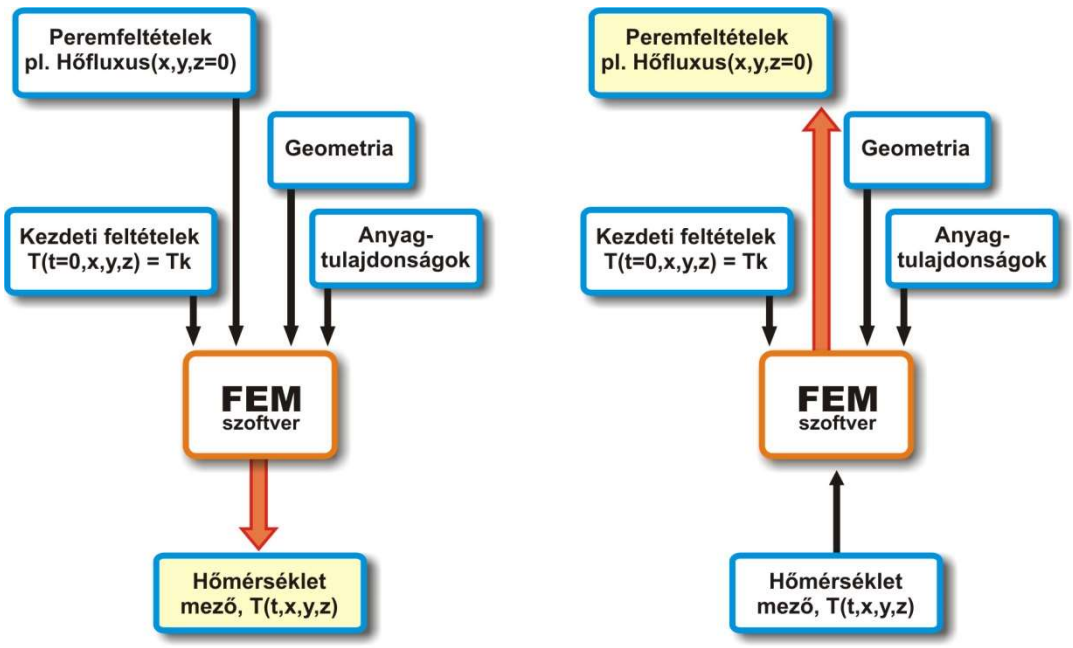
A hővezetési Fourier egyenlet megoldására a Smith féle explicit, Véges Differencia Módszert (Finite Difference Method) alkalmaztam. [4]

3.2. Inverz hőtani modellezés

Egy hűtési vagy hevítési folyamatnak kitett testben a hőmérséklet-eloszlás változását az idő és helykoordináta függvényében úgy lehet meghatározni, hogy előállítjuk a hővezetés Fourier-egyenletének (2.1 egyenlet) megoldását a szokásos harmadfajú (konvekciós) peremfeltétel mellett (2.3 egyenletek). A hőátadási együttható és az egyéb más fizikai tulajdonságok (hővezetési tényező, fajhő) jelentősen függnek a hőmérséklettől és a lehűlési folyamat közben kialakuló, átalakuló fázisok típusától, minőségétől. Ebből adódóan az ilyen típusú hőközlési folyamatok matematikai szempontból nem-lineárisnak tekinthetők. A Fourier egyenlet megoldása zárt alakban nem állítható elő, ezért a hőmérséklet-eloszlás meghatározására csak numerikus módszerek alkalmazhatók.

A hőmérsékleteloszlást leíró függvény egyértelműen definiálható, ha a kezdeti és peremfeltételek megfelelően állnak rendelkezésünkre, azaz így garantálható az egyetlen, egyértelmű megoldás létezése. Ebben az esetben „helyesen feltett” problémával állunk szemben, aminek a megoldása a rendelkezésre álló numerikus módszerekkel nagyrészt problémamentes. Ezt „Direkt modellezési eljárásnak” nevezzük. (2.4 ábra)

Alifanov megfogalmazása [5] szerint a helyesen feltett probléma tulajdonsága, hogy az **ok** (hőátadási együttható vagy hőfluxus) ismeretében, numerikus módszerrel egyértelműen meghatározható az **okozat** (vagyis a kialakuló hőmérsékleti mező).



Direkt modellezés Inverz modellezés

2.4 ábra A Direkt és az Inverz modellezés elvi ábrája

A termikus peremfeltételek, például a Hőátadási Együttható becslésére ezzel szemben a hőmérsékletmező ismeretében van lehetőség, amelyet sok esetben csak méréssel lehet meghatározni. A „Direkt" modell „**ok-okozati**" sorrendjével szemben, a probléma-felvetés inverz jellegű, vagyis az **okozat** ismeretében kell következtetnünk az **okra**. Ezt „rosszul feltett" (ill-posed) problémának nevezik [2]. Ebben az esetben a megoldandó feladat a peremfeltételek rekonstruálása (hővezetési tényező, felületi hőfluxus meghatározása) a próbatestben méréssel meghatározott (input) hőmérséklet-eloszlás ismeretében. A rosszul feltett problémák tulajdonsága, hogy a megoldás előállítása rendkívül sok számítást igénylő feladatot jelent, mert a direkt módszerek közvetlen alkalmazása nem vezet eredményre [5]. Ilyen esetekben Inverz módszerről beszélünk. (2.4 ábra).

A méréssel meghatározott hőmérsékletgörbék mellett a munkadarab hőtani-szimulációs modellje elengedhetetlen az inverz számítási módszerhez. A próbatesten elhelyezett termoelemek jelének feldolgozásával a határfelületek hőátadási „térképe" az idő és a hely-koordináták függvényében állítható elő. Az alkalmazott számítási eljárások, amelyek a hőáram térképek becslésére alkalmazandók, speciális numerikus módszereken alapulnak. A feladat egy nem-lineáris, tranziens inverz hőközlési probléma megoldását feltételezi (IHCP, Inverse Heat Conduction Problem).

A dolgozat célkitűzésének teljesítéséhez szükséges inverz hőtani algoritmus működését a 2.5 ábra szemlélteti. A Hőátadási Együttható számítására használt rendszerek az általánosan elfogadott módszerek [2,3] szerint a következő lépésekből állnak:

1. lépés A hőátadási folyamat során a vizsgált munkadarab p darab pontjaiban a lehűlési görbék $(T_i^m, i = 1 \dots p)$ felvétele a $[t_s, t_f]$ intervallumban.
2. lépés Az idő függvényében definiált hőátadási együtthatók $(h_i(t,r), i = 1 \dots p)$ meghatározása a $[t_s, t_f]$ intervallumban.
3. lépés A lehűlési görbék számítása $(T_i^c, i = 1 \dots p)$ $h_i(t,r)$ alapján.
4. lépés A mért és számított lehűlési görbék összehasonlítása, S számítása.

5. lépés Ha a különbség a megadott tolerancia határon kívül esik, akkor $h_i(t)$ módosítása $\Delta h_i(t)$-val és visszatérünk a 2. lépéshez.

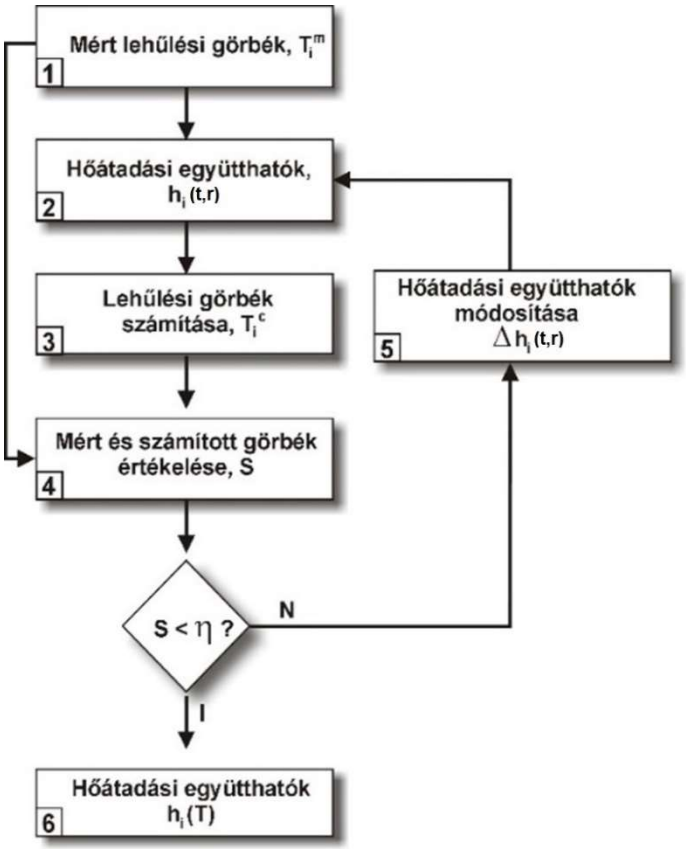
6. lépés Ha a közelítés megfelelő, akkor a számítás végeredménye a teljes hőmérséklet ciklus ismeretében a hőátadási együtthatók, mint a felületi hőmérséklet függvényei $h_i(T)$.

A 2-5 lépések iteratív végrehajtásával arra keressük a választ, hogy mely $h_i(t,r)$ függvény mellett a legkisebb a mért és számított hőmérsékleti minták közötti eltérés, azaz az S értéke. Az S értékének a minimalizálása egy szélsőérték keresési feladatra vezethető vissza, ahol S minimumát keressük, és S a (2.4 egyenlettel) számítható ki:

$$S = \sum_{i=1}^{p} (T_i^c - T_i^m)^2 \qquad (2.4)$$

Az iterációs folyamat (2.5 ábra) kulcsa és egyben meghatározó lépése az 5. lépésében végrehajtott $h_i(t,r)$ módosítása. Az újabb iterációban felhasznált $\Delta h_i(t,r)$ az alkalmazott optimalizálási algoritmus függvénye. Kézenfekvőnek tűnik, hogy abban az esetben, ha a Hőátadási Együttható függvényt leíró paraméterek száma csekély (<10), akkor a közismerten hatékony, egyszerűen implementálható optimalizálási módszert (pl. Simplex vagy valamely gradiens módszert) alkalmazzuk. A szakirodalmi források [3,4,5] azonban egyértelműen utalnak arra, hogy az előbb említett optimalizálási algoritmusok hajlamosak a lokális szélsőértékekben „ragadni", azaz gyakran nem alkalmasak a problématérben a globális minimum felfedésére. Ennek oka egyrészt az algoritmusok működésében keresendő. Másrészt, az optimalizálás eredménye a számítások indításánál megadott kiinduló értéktől, azaz a 2. lépésben inicializált $h_i(t,r)$ kezdeti beállításától erősen függ. Mint ismeretes, számos optimalizálási módszer érzékeny arra, hogy a problématér mely pontjáról indul a számítás [4,5]. Ennek a nem kívánatos jelenségnek az elkerülése érdekében célszerű olyan robosztus optimalizálás módszert választani, mely nem érzékeny a kezdeti állapotra és lehetőség szerint gyorsan konvergál.

A következő fejezetben röviden ismertetek néhány heurisztikus optimalizációs algoritmust és részletesen bemutatom a probléma megoldására kiválasztott raj alapú Tűzijáték algoritmust és variánsait.



2.5 ábra Az inverz algoritmus lépései

3.3. Optimalizációs algoritmus

Ebben a fejezteben olyan heurisztikus algoritmusokról ejtek szót, amelyek alkalmasak lehetnek vagy már alkalmazták is a 3.2 fejezetben felvázolt inverz problémához hasonló feladatok megoldására. Röviden ismertetem a Genetikus Algoritmus és a Részecske Raj Optimalizáció működését, ezután részletesen kitérek a Tűzijáték Algoritmusra és variánsaira.

3.3.1. Genetikus Algoritmus

A Genetikus Algoritmus (GA, Genetic Algorithm) a darwini evolúció elméleten alapul és az evolúciós algoritmusok osztályába tartozik. Működésének lényege, hogy a lehető legjobb tulajdonságokkal rendelkező egyed létrehozására törekszik az evolúció során. Ezt olyan biológiai mechanizmusok segítségével teszi, mint a mutáció, keresztezés és a természetes szelekció. A kromoszómák reprezentálják a probléma megoldásának variációit. Minden egyed saját kromoszómával rendelkezik, illetve kiválasztásra kerülhet az úgynevezett „mating poolba”, ahol más tagok génjeinek keresztezésével új egyedeket hoznak létre. Mindez a reprodukciós szakasz során történik. A génmutáció biztosítja, hogy a populáció génállománya is folyamatosan módosuljon a diverzitás fenntartása érdekében, mert különben a populáció tagjai csak egymás között cserélgetnék a génjeiket. A szelekciós mechanizmus választja ki a populációból azokat az egyedeket, amelyek tovább élhetnek, ezzel tovább örökítve a saját génjeiket a következő generációba. A szelekció fontos szerepet játszik az algoritmusban, mivel annak konvergenciájának sebességét befolyásolja. Ezért ennek a mechanizmusnak az implementációjára több stratégia is született, mint például a rulettkerék kiválasztás. Fontos szabálya, hogy a jobb tulajdonságokkal rendelkező egyedek nagyobb valószínűséggel szaporodnak és a túlélési esélyük is kedvezőbb a társaikhoz képest. A verseny alapú kiválasztásnál egy egyedet több alkalommal versenyeztetünk a többi egyeddel, majd megszámoljuk, hogy hányszor nyert. Ez a szám adja a rangját, amely alapján eldöntjük, hogy tovább mehet-e a következő generációba. Noraini Mohd Razali és John Geraghty arra a következtetésre jutottak a kutatásuk alapján [6], hogy a verseny alapú szelekció a kisebb, a rulettkerék szelekció pedig a nagyobb paraméterszámú optimalizációs problémák megoldására alkalmas. Kellő pontosságú eredmény érdekében a kétdimenziós inverz hőátadási probléma paraméterszáma is nagy, ezért ehhez rulettkerék szelekciót érdemes használni. Hasonló Inverz Hőközlési Probléma megoldásához Czél Balázs és Gróf Gyula is alkalmazta az algoritmust [7]

[8]. Az algoritmust ezenkívül olyan komplex problémák megoldására is alkalmazzák, mint például a mellrák szűrése [9].

3.3.2. Részecske-raj optimalizáció

A Részecske-raj optimalizáció (PSO, Particle Swarm Optimization) egy biológiailag inspirált raj alapú algoritmus, amelynek alapja a madarak és hal rajok mozgása. Kennedy és Eberhart 1998-ban mutatta [10] be az algoritmust. Lényege, hogy a rendszer egy rajt szimulál, melynek tagjai a részecskék. A részecskék pozíciójuk változtatásával folyamatosan együtt mozognak és keresik az optimumot a keresési térben. Minden részecske koordinátái egy lehetséges megoldását reprezentálják a problémának. A Genetikus Algoritmustól eltérően nincsenek generációk és a részecskéket a térbeli koordinátáik határozzák meg. A keresés során mindig ugyanazok a részecskék térképezik fel a teret, amelyeket a kezdésnél létrehoztunk. Minden x_i részecskének van egy $x_i = (x_i^1, x_i^2, \dots x_i^D)$ pozíciója és $v_i = (v_i^1, v_i^2, \dots v_i^D)$ sebessége (ahol *D* a problématér dimenziója). Az algoritmus megfelelő működéséhez biztosítani kell, hogy két pozíció különbsége egy sebesség legyen. A sebességet meg lehessen szorozni skalár értékkel és össze lehessen adni két sebességet, illetve egy pozícióhoz hozzá lehessen adni egy sebességet. A rajnak van egy g^{opt} globális optimuma, amelyet mindenki lát és a részecskéknek egy saját p^{opt} lokális optimuma. A részecskék sebességét a következő képlet alapján változtatjuk minden iterációban $v_i^{t+1} = v_i^t + c_1 r_1 (p_i^{opt} - x_i^t) + c_2 r_2 (g_i^{opt} - x_i^t)$ ahol, t az iteráció száma, $c_1 c_2$ konstansok, $r_1 r_2$ pedig véletlen számok. A kiszámított sebességet hozzáadjuk a pozíciójukhoz, így ezzel megkapjuk a következő pozíciójukat. Ha egy részecske az eddigieknél jobb megoldást talál, akkor a többi is elkezd felé haladni úgy, hogy a saját maguk által felfedezett legjobb lokális optimumot is figyelembe veszik. Az algoritmus egyszerűsége és hatékonysága miatt előszeretettel használják inverz problémák megoldására. Fung-Bao Liu [11] kristályból készült hengeres testen létrejövő hőátadás becslésére használta, emellett P. Dousti, A.A. Ranjbar, M. Famouri és A. Ghaderi [12] kétdimenziós Inverz Hőközlési Probléma megoldására alkalmazta.

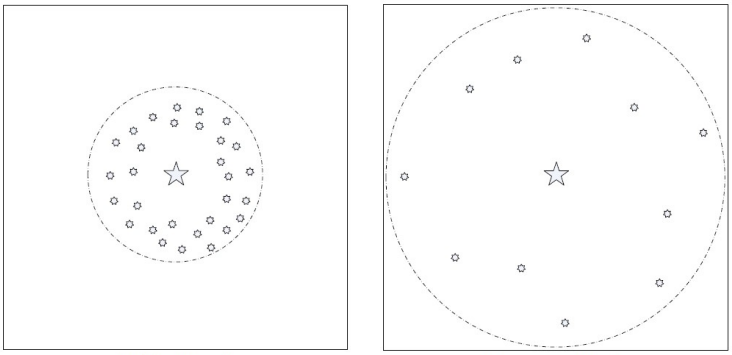
3.3.3. Tűzijáték Algoritmus

A dolgozatomban az Adaptív Tűzijáték Algoritmus módszert alkalmaztam. Ennek a heurisztikának a bemutatásához azonban elengedhetetlen az eredeti [6] és egy továbbfejlesztett változat [7] ismertetése, ezért mind a három algoritmusról külön ejtek szót. Az eredeti

algoritmussal igyekszem szemléltetni a módszer alapgondolatát és működését. Ezután ismertetem az Fejlesztett Tűzijáték Algoritmus [13] variánst, ami az operátorok tekintetében tér el jelentősen az eredeti módszertől. Ebben a változatban javították ki ugyanis az eredeti algoritmus hibáit, illetve az összes később publikált variáns ebből származik. Végül bemutatom az Adaptív Tűzijáték Algoritmus [14] változatot, amely kisebb, de nagyon hatékony módosítást tartalmaz az un. amplitúdó operátorban.

Ying Tan és Yuanchun Zhu 2010-ben publikálta elsőként azt a levegőbe fellőtt tűzijátékok viselkedését imitáló algoritmust [6], mely a sztochasztikus, raj alapú optimalizálási módszerek csoportjába tartozik.

A valós (levegőbe fellőtt) tűzijátékoknál a hordozó rakéta (ez a tűzijáték, vagy Firework) repülése végén felrobban és változatos mintát rajzol az égre. A „mintázatokat” végső soron a felrobbanó rakéta tölteteinek repülési pályája adja, azaz a rakéta felrobbanásának helye (mint kiinduló pont) és rakétában lévő töltet ballisztikus útvonalának végpontja közötti szakasz. A ballisztikus végpontot ebben az esetben „csillagnak” nevezzük. A tűzijáték mintázat változatosságát a robbanás nagysága (amplitúdója) és a csillagok számossága határozza meg. A paraméterek alapján két kategóriába lehet osztani a robbanásokat. „Jó” robbanásnak nevezzük azt a robbanást, amelynek amplitúdója viszonylag kicsi és viszonylag sok csillaggal rendelkezik. Az elnevezés abból származik, hogy a fényes és elkápráztató tűzijátékok sokkal látványosabbak az emberi szem számára. Ezzel ellentétesen „rossz”-nak nevezzük azt a robbanást, amelyeknek nagy amplitúdója és kevés csillagja van. Az optimalizáció során mindegyik tűzijáték csillagainak száma és robbanásának nagysága attól függ, hogy az összes többi tűzijátékhoz képest az éppen aktuális milyen távolságra van a globális szélső értéktől. Minél közelebb van egy tűzijáték a globális optimumhoz, akkor annál nagyobb a csillagainak száma és kisebb a robbanásának amplitúdója. Természetesen a két típus között nincs egy jól elkülöníthető határ: pusztán az alap koncepció szemléltetésére szolgál a kategorizálás.

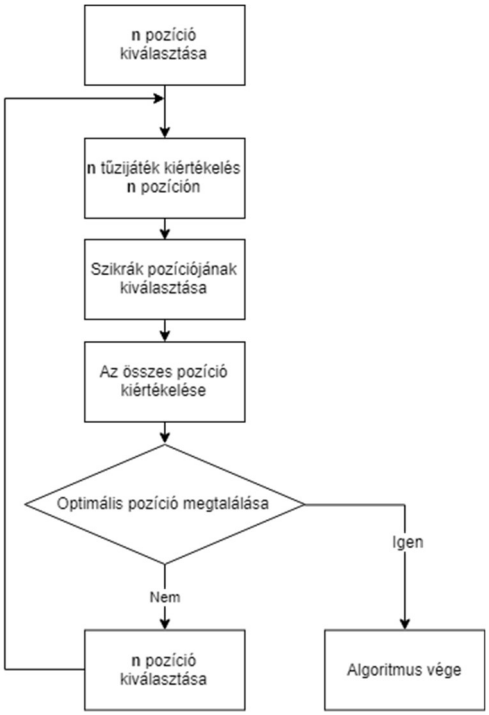


3.1 ábra Robbanások minősítése (forrás: [15])

3.3.3.1. Eredeti Tűzijáték Algoritmus

A 2010-ben publikált, eredeti Tűzijáték Algoritmus működése négy lépésre és öt operátorra bontható (3.1.ábra). Az **első** lépés az összesen n darab tűzijáték inicializációja, vagyis valamilyen (általában random) módon a kezdeti paramétereik meghatározása. A **második** lépésben a tűzijátékok kiértékelése történik, más szóval az optimalizálási térben „felrobbanó" rakéta által meghatározott helyen a célfüggvény értéknek (esetünkben S, 2.4 egyenlet), azaz a „fitness"-nek a kiszámítása. Az ismert „fitness" alapján, így a **harmadik** lépésben már kiszámítható az egyes tűzijátékokhoz tartozó csillagok száma, illetve a robbanások nagysága. A csillag populációnak maximum m darab csillag-egyede lehet. Ezt az összesen m darab csillagot az algoritmus a „fitness" értékek függvényében osztja szét az egyes tűzijátékok között (3.1 egyenlet). A robbanások nagysága, azaz Amplitúdója az $\hat{A}$ kontroll paraméter és a „fitness" érték alapján kerül kiszámításra (3.3 egyenlet). Ugyancsak a **harmadik** lépésben az adott tűzijátékhoz m_i darab csillag pozícióját határozzuk meg az A_i amplitúdón belül, az *1. Algoritmus*t követve. A csillag populáció változatosabbá tételének céljából $\hat{m}$ darab **speciális csillagot** szükséges létrehozni (*2. Algoritmus*), amelyek térbeli helyzete néhány véletlenszerűen kiválasztott, előzetesen generált csillag és a legjobb (a globális szélsőértékhez legközelebbi fitness értékű) tűzijáték, mint két térbeli pont által meghatározott egyenesen helyezkednek el. A speciális csillagok pozícióinak meghatározásához a normál eloszlást követő véletlengenerátort alkalmazzuk és emiatt ezeket a csillagokat Gauss csillagoknak is nevezik. A **negyedik** lépésben végül az összes csillag kiértékelésére és a legjobb csillag kiválasztására kerül sor, amelyet a következő iterációban alkalmazunk referencia értékként. Az **ötödik** lépésben a leállási feltételt ellenőrizzük, amennyiben a legjobb csillag fitness értéke a

meghatározott tartományban van (vagy elértük a maximális iteráció számot), úgy leállítjuk a számítást, ellenkező esetben új iterációt indítunk.



3.2 ábra Tűzijáték algoritmus (forrás: [15])

A továbbiakban az operátorok működését részletezem.

Az **EXP** operátor a csillagok számának kiszámítására hivatott. A Tűzijáték Algoritmust általános optimalizációs problémák megoldására tervezték, ahol keressük a minimumot $f(x) \in \mathbb{R}, x_{min} \leq x \leq x_{max}$ tartományban. Az $x = x_1, x_2, \dots, x_d$ paraméter vektort jelöl, az $f(x)$ a célfüggvényt. A tartomány alsó és felső korlátját x_{min} és x_{max} jelöli. Egy tűzijátékot x_i jelöl. Az x_i tűzijáték által generált csillagok száma pedig a következő:

$$s_i = m \cdot \frac{y_{max} - f(x_i) + \varepsilon}{\sum_{j=1}^{n} \left(y_{max} - f(x_j)\right) + \varepsilon} \quad , (3.1)$$

ahol m egy paraméter, ami szabályozza, hogy összesen hány csillag legyen a populációban. $y_{max} = \max\left(f(x_i)\right) (i = 1,2, \dots, n)$ a legrosszabb tűzijáték „fitness” értéke. Nullával való osztás elkerüléséhez a nevezőhöz és a számlálóhoz is hozzá adódik az ε konstans. A nevezőben a tűzijátékok távolsága a legrosszabb tűzijátéktól van összegezve. Annak érdekében, hogy ne

legyenek túl sok vagy túl kevés csillaggal rendelkező tűzijátékok, az alábbi korlátok vannak meghatározva:

$$\widehat{s_i} = \begin{cases} round(a \cdot m) & ha\ s_i < a \cdot m \\ round(b \cdot m) & ha\ s_i > b \cdot m,\ \ a < b < 1 \\ round(s_i) & egyébként \end{cases}, (3.2)$$

ahol a alsó és b felső korlátot kijelölő paraméterek, s_i az adott tűzijáték csillagainak száma.

Az **AMP** operátorral a robbanások nagyságát (amplitúdóját) számítjuk ki:

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{min} + \varepsilon}{\sum_{j=1}^{n}(f(x_j - y_{min}) + \varepsilon}, (3.3)$$

ahol $\hat{A}$ paraméter a robbanás maximumát jelöli. $y_{min} = min(f(x_i))\ (i = 1,2, \dots, n)$ a legjobb tűzijáték fitness értéke. A tört nevezőjében a tűzijátékok távolsága a legrosszabb tűzijátéktól van összegezve.

A robbanás nagyságának és a csillagok számának ismeretében a következő algoritmus szerint lehet a csillagok helyzetét kiválasztani:

1. Algoritmus: Csillagok elhelyezése

Csillagok inicializálása $\tilde{x}_j = x_i$;
$z = round(d \cdot rand(0,1))$;
Véletlenszerűen z db dimenzió kiválasztása $\tilde{x}_j$-ből;
Eltolás kiszámítása: $h = A_i \cdot rand(-1,1)$;
ciklus $\tilde{x}_k^j \in \{kiválasztott\ dimenziók\ \tilde{x}_j - ből\}$
  $\tilde{x}_k^j = \tilde{x}_k^j + h$;
  ha $\tilde{x}_k^j < \tilde{x}_k^{min}\ vagy\ \tilde{x}_k^j > \tilde{x}_k^{max}$ **akkor**
    $\tilde{x}_k^j$ elhelyezése a tartományon belül: $\tilde{x}_k^j = \tilde{x}_k^{min} + |\tilde{x}_k^j|\ \%\ (\tilde{x}_k^{max} - \tilde{x}_k^{min})$;
  elágazás vége
ciklus vége

A paraméter vektorból véletlenszerűen kiválasztjuk $\tilde{x}_j$ koordinátát és egy véletlenszerűen 0 és 1 között kiválasztott értékkel eltoljuk. Ha az új koordináta a vizsgált paraméter téren kívülre esik, akkor visszahelyeződik a helyes tartományba.

Ehhez a **MAP** operátorhoz tartozó 3.4 egyenlet használható:

$$\tilde{x}_k^j = \tilde{x}_k^{min} + \left|\tilde{x}_k^j\right| \,\%\, \left(\tilde{x}_k^{max} - \tilde{x}_k^{min}\right), (3.4)$$

ahol az $\tilde{x}_k^j$ koordináta által felvehető legkisebb és legnagyobb értéket $\tilde{x}_k^{max}$ és $\tilde{x}_k^{min}$ jelöli. Az így kapott koordináta az újonnan létrejött csillag koordinátája. Az a koordináta, amely nem került kiválasztásra, annak ugyanaz az „őse" által inicializált értéke marad.

A **GAU** operátor a speciális Gauss csillagok létrehozásáért felelős, amelyek a populáció diverzitását hivatottak erősíteni. A Gauss csillagok koordinátáit a 2. algoritmus alapján számítjuk ki.

2. Algoritmus: Gauss csillagok elhelyezése

Gauss csillagok inicializálása $\tilde{x}_j = x_i$;
$z = round\left(d \cdot rand(0,1)\right)$;
Véletlenszerűen z db dimenzió kiválasztása $\tilde{x}_j$-ből;
Együttható kiszámítása $g = Gaussian(1,1)$;
Eltolás kiszámítása: $h = A_i \cdot rand(-1,1)$;
ciklus $\tilde{x}_k^j \in \{kiválasztott\ dimenziók\ \tilde{x}_j - ből\}$
  $\tilde{x}_k^j = \tilde{x}_k^j \cdot g$;
  ha $\tilde{x}_k^j < \tilde{x}_k^{min}\ vagy\ \tilde{x}_k^j > \tilde{x}_k^{max}$ **akkor**
    $\tilde{x}_k^j$ elhelyezése a tartományon belül: $\tilde{x}_k^j = \tilde{x}_k^{min} + \left|\tilde{x}_k^j\right| \,\%\, \left(\tilde{x}_k^{max} - \tilde{x}_k^{min}\right)$;
  elágazás vége
ciklus vége

Az együttható értéke, amellyel a származtatott koordinátát szorozzuk, egy Gauss eloszlású függvényből származik. A függvény várható értéke $\mu = 1$ és szórása $\sigma = 1$.

A **SEL** operátor a következő iteráció tűzijátékainak kiválasztásáért, azaz a tűzijáték rakéták felrobbanási koordinátáinak meghatározásáért felelős. Az adott iteráció végén kiválasztásra kerül a legjobb csillag, mely a következő iterációban lévő megoldás populációnak (a rakéták és a csillagok csoportja) is a tagja lesz. A legjobb csillagon kívül a következő iterációba $n-1$ csillag kerül be, amelyeket a többiektől való távolságuk alapján kell kiválasztani. Minél

távolabb van egy csillag a többitől, annál nagyobb eséllyel kerül kiválasztásra. Az x_i csillag és egy másik csillag közötti távolság számítása a 3.5 egyenlet alapján történik:

$$R(x_i) = \sum_{j \epsilon K} d(x_i, x_j) = \sum_{j \epsilon K} \|x_i - x_j\| \quad , (3.5)$$

Ahol K az összes jelenleg ismert tűzijátékok és a csillag száma. Az x_i csillag vagy tűzijáték kiválasztásának valószínűségét a 3.6 egyenlettel határozzuk meg:

$$p(x_i) = \frac{R(x_i)}{\sum_{j \epsilon K} R(x_j)} \quad (3.6)$$

Végül az összes operátor felhasználásával maga a Tűzijáték Algoritmus struktúrája a következő:

3. Algoritmus: Tűzijáték algoritmus

Tűzijátékok véletlenszerű inicializálása;
ciklus amíg $leállási\ feltétel\ = hamis$
  Tűzijátékok kiértékelése a célfüggvény alapján;
  ciklus $i = 1 - től\ n - ig$
    Az x_i tűzijátékhoz tartozó csillagok számának kiszámítása a (3.1) alapján;
    Az x_i tűzijátékhoz tartozó robbanás nagyságának számítása a (3.3) alapján;
    Az x_i tűzijátékhoz tartozó csillagok létrehozása az 1. Algoritmus alapján;
  ciklus vége
  ciklus $k = 1 - től\ \hat{m} - ig$
    Egy x_j tűzijáték véletlenszerű kiválasztása;
    Gauss csillag létrehozása a kiválasztott tűzijátékhoz a 2. Algoritmus alapján;
  ciklus vége
  Az összes csillag kiértékelése és a legjobb csillag eltárolása a következő iterációra;
  $n - 1$ darab csillag vagy tűzijáték véletlenszerű kiválasztása a (3.6) alapján
ciklus vége

Az algoritmus működéséhez a célfüggvény kiértékelése $(n + m + \hat{m})$ alkalommal történik meg minden iterációban. Feltételezve, hogy az $f(x)$ függvény optimuma T iteráció alatt található meg, akkor az algoritmus kiértékeléseinek a száma $T \cdot (n + m + \hat{m})$.

3.3.3.2. Fejlesztett Tűzijáték Algoritmus

Az eredeti Tűzijáték Algoritmus hatékonyan működik olyan célfüggvényeken, amelyeknek az optimuma az origóhoz közel helyezkedik el [7]. Ellenben, ha a függvény optimuma távol van az origótól, akkor a konvergálási sebesség jelentősen lecsökken [8]. Az eredeti algoritmus sok számítást igényel, mely jelentősen növeli az optimum megtaláláshoz szükséges időt Az előnytelen jellemzők hatásának csökkentése érdekében az 5 operátor módosítására került sor a Fejlesztett Tűzijáték Algoritmus változatban olyan módon, hogy az algoritmus működésének alap koncepciója ugyanaz maradt. Változtatás tehát az **AMP, EXP, MAP, GAU** és **SEL** operátorokban történt.

Az eredeti **AMP** operátor úgy működik, hogy a legjobb tűzijáték robbanásának mérete olyan kicsi, hogy a csillagainak elhelyezkedése közelítőleg megegyezik. Ez felesleges számításokat eredményez, amelyek kevésbé hatékonyak az optimumkeresés szempontjából Az újabb operátorban bevezetésre került a robbanás méretére vonatkozó minimum értéket kijelölő változó. Ha egy robbanás mérete kisebb, mint ez a minimum, akkor ez az érték kerül beállításra a 3.7 képlet alapján:

$$A_i^k = \begin{cases} A_{min}^k & ha\ A_i^k < A_{min}^k \\ A_{min}^k & egyébként \end{cases}, (3.7)$$

Ez a minimum folyamatosan csökken az iterációk előrehaladtával. A változó kezdeti értékét és végső értékét A_{init} és A_{final} jelöli. A célfüggvény kiértékelésének maximumát ($evals_{max}$) is fel kell használni a robbanás méret minimumának kiszámításához. Ez a minimum az iterációk során nem lineáris módon csökken (3.8 egyenlet)

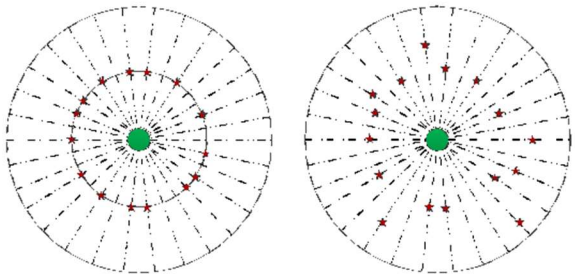
$$A_{min}^k(t) = A_{init} - \frac{A_{init} - A_{final}}{evals_{max}} * \sqrt{(2 * evals_{max} - t) * t}\,. \quad (3.8)$$

Egy csillag koordinátáinak eltolása ugyanazzal az értékkel történik az eredeti **EXP** operátorban. Ez az optimalizációs tér nem feltétlenül hatékony „feltérképezését” eredményezi, mert a koordináták az összes dimenzióban ugyanakkora értékkel változnak meg. A diverzitás növelése érdekében az új operátor a csillag koordinátáit minden dimenzióban más értékkel változtatja meg (*4. Algoritmus*), az alábbiak szerint:

4. Algoritmus: Csillagok elhelyezése az EFWA-ban

Csillagok inicializálása $\tilde{x}_j = x_i$;
$z^k = round\big(rand(0,1)\big), \qquad k = 1,2,\ldots..d$;
ciklus $\tilde{x}_k^j \in \{kiválasztott\ dimenziók\ \tilde{x}_j - ből\}$, ahol $z^k == 1$
  Eltolás kiszámítása: $h = A_i \cdot rand(-1,1)$;
  $\tilde{x}_k^j = \tilde{x}_k^j + h$;
  ha $\tilde{x}_k^j < \tilde{x}_k^{min}\ vagy\ \tilde{x}_k^j > \tilde{x}_k^{max}$ **akkor**
    $\tilde{x}_k^j$ elhelyezése a tartományon belül: $\tilde{x}_k^j = \tilde{x}_k^{min} + \left|\tilde{x}_k^j\right| \%\ \left(\tilde{x}_k^{max} - \tilde{x}_k^{min}\right)$;
  elágazás vége
ciklus vége

Ez az operátor jobb „felfedező” képességet eredményez [7], mivel nem csak egy kör kerületén helyezkednek el a csillagok.



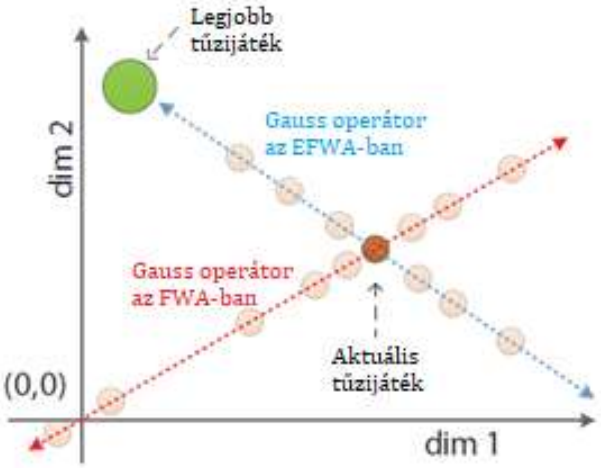
(a) FWA (b) EFWA

3.3 ábra Az új operátor látható hatása a csillagok elhelyezkedésére (forrás: [13])

Az eredeti **MAP** operátor felelős részben azért, hogy az origó közelében jön létre rengeteg csillag és tűzijáték. Ez azért történik ilyen módon, mert a populáció egyedei sokszor átlépik a tartomány korlátjait. Ha a keresési tartomány kiegyenlített, akkor ezek a csillagok az origó közelében kapnak értéket és annak környezetében tömörülnek. Az új operátor (3.9 egyenlet) egységes eloszlást követve véletlenszerűen helyezi el a csillagokat az egész tartományban. Így a tartományon belül akárhol megjelenhetnek anélkül, hogy az origó kis környezetében csoportosulnának.

$$\bar{X}_i^k = X_{min}^k + rand * \left(X_{max}^k - X_{min}^k\right), \quad (3.9)$$

Az eredeti Tűzijáték Algoritmus előnye, hogy csekély számú iteráció után is megtalálja az olyan optimumot, amely az origótól kis távolságra helyezkedik el. Ezt a hatékonyságot a GAU és a MAP operátor összhatása okozza, ahogyan az a 3.4 ábrán látható. A Gauss csillagok az origó és a véletlenszerűen kiválasztott tűzijátékok tengelyén jönnek létre és sokszor nulla közeli értéket vesznek fel a koordinátáik.



3.4 ábra Gauss csillagok pozícionálása (forrás: [13])

Azért, hogy a Gauss csillagok egyenletesen helyezkedjenek el a keresési térben, az új **GAU operátor** a legjobb tűzijáték és egy véletlenszerűen kiválasztott tűzijáték között elhelyezkedő tengelyen hozza létre a csillagokat. A 3.10 egyenletben az X_B^k a legjobb tűzijátékot jelöli, a $\hat{X}_i^k$ pedig a kiválasztott tűzijátékot:

$$\hat{X}_i^k = \hat{X}_i^k + \left(X_B^k - \hat{X}_i^k\right) * e, (3.10)$$

Így a Gauss csillagok elhelyezése az alábbi algoritmus szerint történik:

5. Algoritmus: Gauss csillagok elhelyezése az EFWA-ban

Gauss csillagok inicializálása $\tilde{x}_j = x_i$;
$z^k = round\left(rand(0,1)\right),\ k = 1,2,\ldots..d$; $\tilde{x}_j$-ből;
Együttható kiszámítása $e = Gaussian(0,1)$;
ciklus $\tilde{x}_k^i \in \{kiválasztott\ dimenziók\ \tilde{x}_j - ből\}$, ahol $z^k == 1$
  $\hat{X}_k^j = \hat{X}_k^i + \left(X_k^b - \hat{X}_k^i\right) \cdot e$
  ha $\tilde{x}_k^j < \tilde{x}_k^{min}\ vagy\ \tilde{x}_k^j > \tilde{x}_k^{max}$ **akkor**
    $\tilde{x}_k^j$ elhelyezése a tartományon belül: $\tilde{x}_k^j = \tilde{x}_k^{min} + \left|\tilde{x}_k^j\right| \%\ \left(\tilde{x}_k^{max} - \tilde{x}_k^{min}\right)$;
  elágazás vége
ciklus vége

A számítási lépések jelentős részét a következő iterációba kerülő csillagok, tűzijátékok kiválasztása teszi ki. Ezért az eredeti **SEL** operátorban a valószínűségi alapon való kiválasztás **elitizmusra** lett lecserélve. Ez azt jelenti, hogy a legjobb elem és $n-1$ véletlenszerűen kiválasztott csillag mehet tovább a következő iterációba. Így az algoritmus kiválasztáshoz kötődő részének futásideje csak lineárisan növekszik a csillagok számának növelésével.

3.3.3.3. Adaptív Tűzijáték Algoritmus

Az optimalizálási tér megfelelő feltérképezéséhez olyan robbanásokra van szükség, amelyek amplitúdói megfelelő ütemben csökkennek. Mivel minden célfüggvény más és más csökkenési „tempót" igényel, ezért ez a feladat nem triviális. Ha a robbanás mérete túl gyorsan csökken, akkor előfordulhat, hogy túl kicsi tartományban keres az algoritmus, miközben a globális optimum távol van ettől a környezettől. Vagy épp ellenkezőleg, az algoritmus nem találja meg az optimumot, mert a robbanás mérete olyan nagy, hogy „átsiklik" felette, mert nem vizsgálja meg elég pontosan a környezetét. Ez a nem kívánt működési jellemző az **AMP** operátor módosításával elkerülhető.

Az előző generációban feljegyzett legjobb tűzijáték robbanásának méretét s^*-gal jelöljük. Az aktuális iterációban, ehhez legközelebb eső tűzijátékot $\hat{s} = argmin(d(s_i, s^*))$ jelzi. A robbanások mérete az aktuális iterációban a 3.11 egyenlet alapján számítható ki

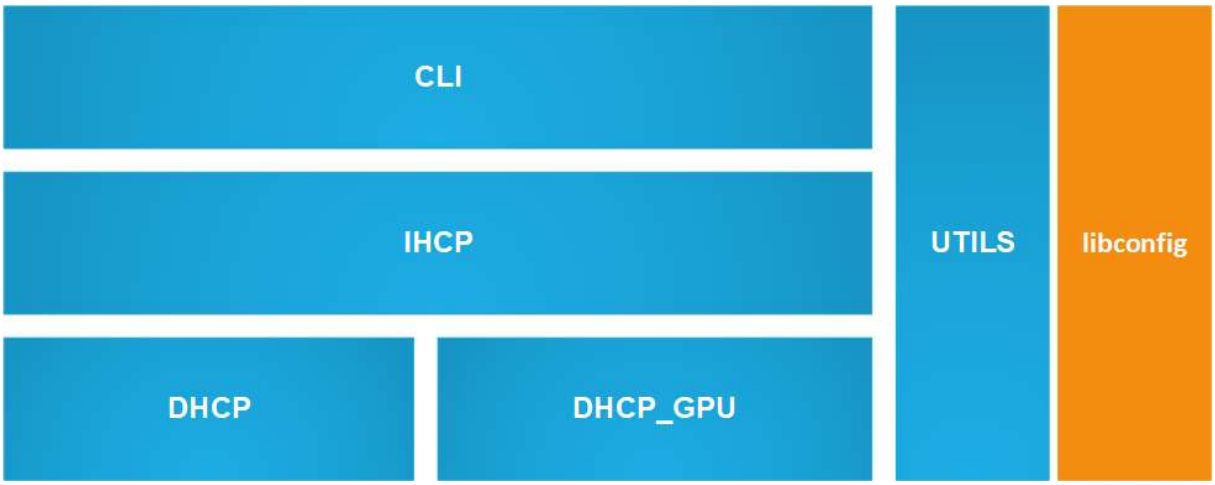
$$A(g+1) = \begin{cases} UB - LB, ha\ g = 0\ vagy\ \forall f(s_i) < f(X) \\ 0.5 * [\lambda * [\![s_i - s^*]\!]_\infty + A(g)],\ egyébként \end{cases}, (3.11)$$

Ahol az UB és LB a keresési tartomány alsó és felső korlátja. $g+1$ a következő iterációt jelöli, λ egy kontroll paraméter, amely az algoritmus felfedező képességét befolyásolja. A szakirodalom szerint érdemes $\lambda = 1.3$ értéket választani ennek a paraméternek. [14]

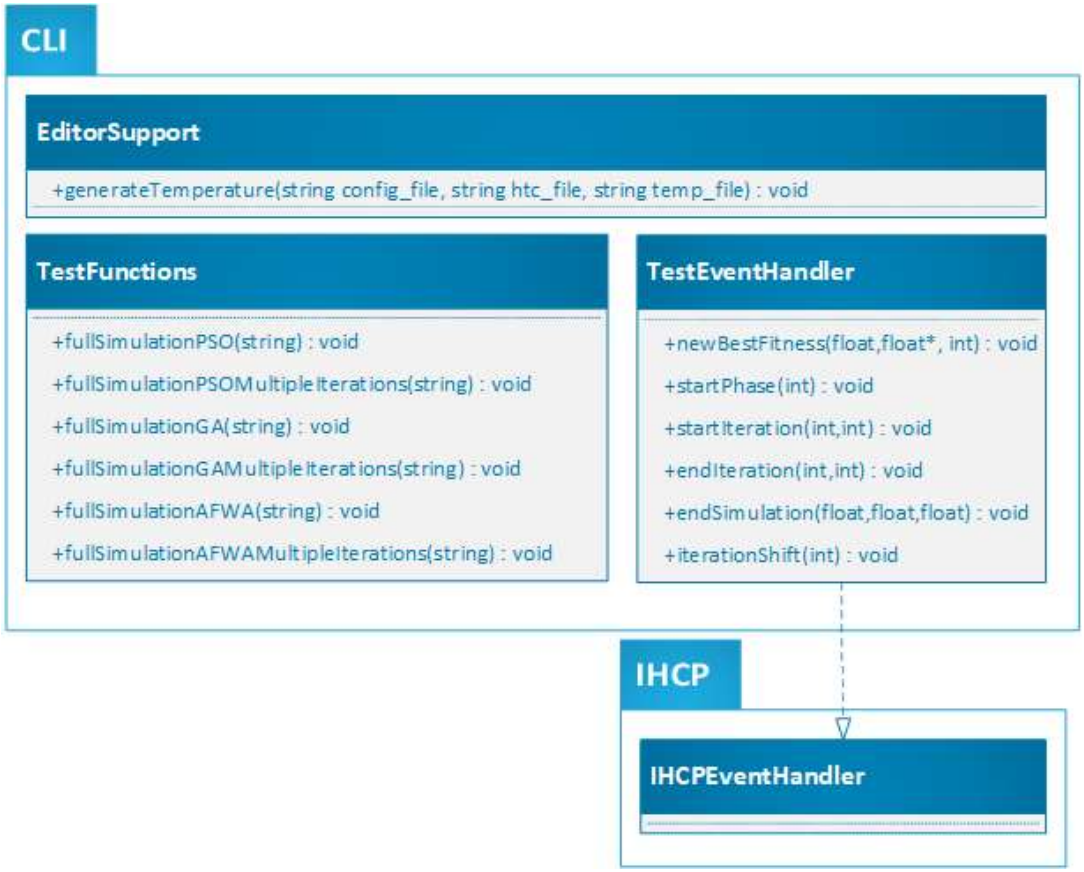
4. Implementáció

4.1. Megvalósított rendszer leírása

A programot C++14-ben implementáltam. A fő alkalmazás a következő négy projektből áll: A CLI konzolos alkalmazásból, az IHCP (dll) dinamikus könyvtárból, amely az optimalizációs algoritmusok implementációit tartalmazza, a DHCP és DHCP_GPU (dll) könyvtárakból, melyek a hőátadási szimulációk CPU és GPU oldali megvalósítását tartalmazzák. A Utils (dll) könyvtár a segédfüggvényeket, a fájlkezelést valósítja meg a libconfig külső könyvtár felhasználásával. A felsorolt rétegek felépítése a 4.1 ábrán látható.



4.1 ábra Az alkalmazás rétegek

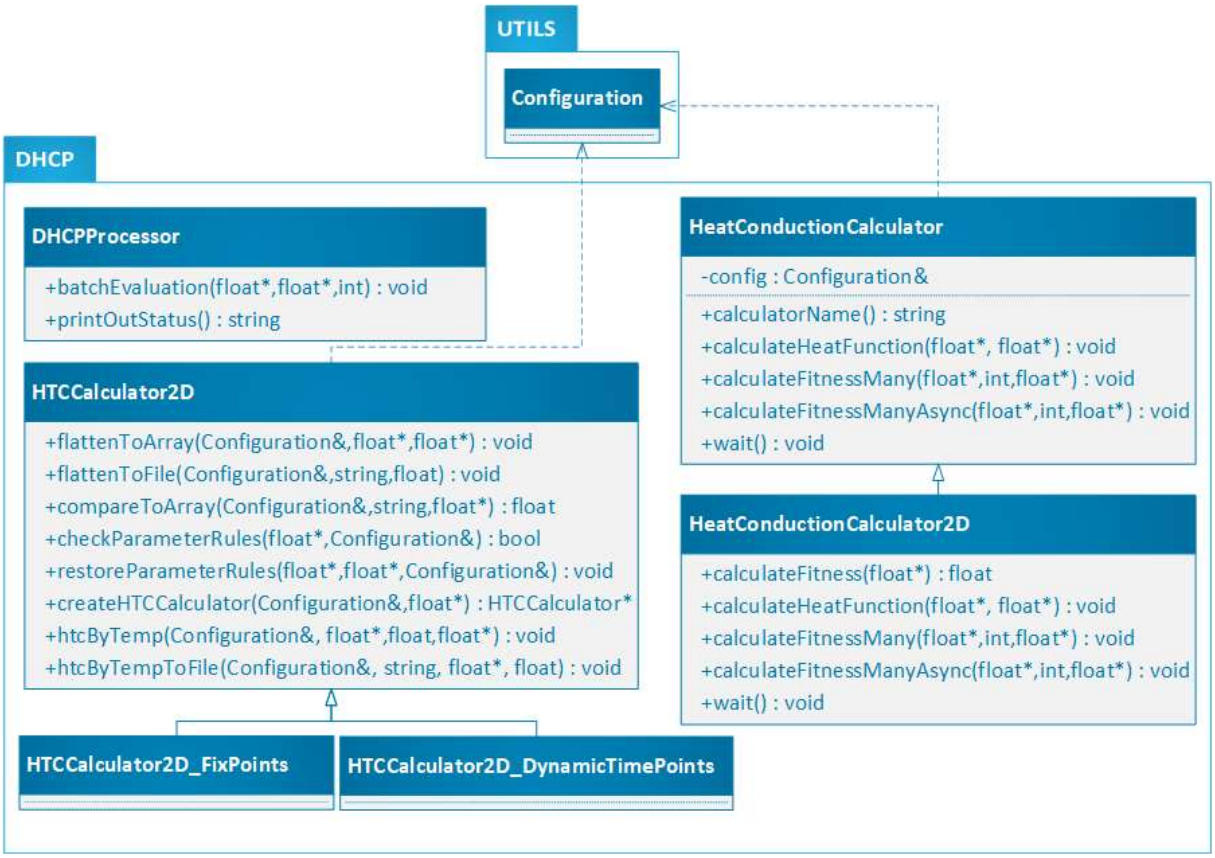


4.2 ábra Konzolos alkalmazás osztálydiagramja

CLI konzolos alkalmazásból indítom a szimulációkat. A TestFunctions osztályban implementáltam a különböző optimalizációs algoritmusokhoz fűződő teszteket. A teszteket konzol paraméter argumentumok beállításán keresztül lehet elindítani. A rendelkezésre álló utasításoknak a listája a 4.1 táblázatban található. A referencia hőmérsékleti görbék generálásához az EditorSupport áll rendelkezésre. A szimulációkhoz és az inverz számításokhoz számos konfigurációs paraméterre van szükség, amelyeket egy (.cfg) kiterjesztésű állományban tárolok. Ezt a későbbiekben fogom részletezni. A TestEventHandler felelős az optimalizáció során kiváltott események kezelésért és az akkor aktuális legjobb paraméterek fájlokba való kiírásáért

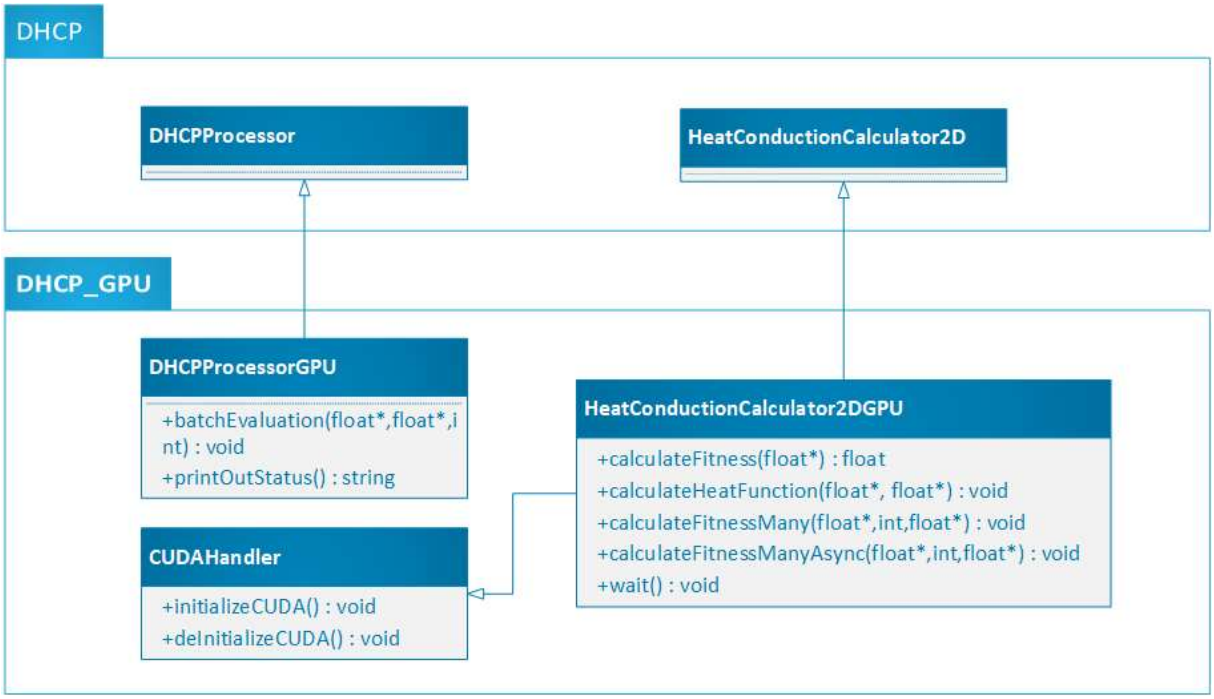
Utasítás		**Paraméterek**
EDITOR		HTC.txt TEMP.txt
TEST-	PSO	*PATH*/pso.cfg
	PSOMI	*PATH* /psomi.cfg
	GA	*PATH* /ga.cfg
	GAMI	*PATH* /gami.cfg
	AFWA	*PATH* /afwa.cfg
	AFWAMI	*PATH* /afwami.cfg

4.1 táblázat Konzolos utasítások listája



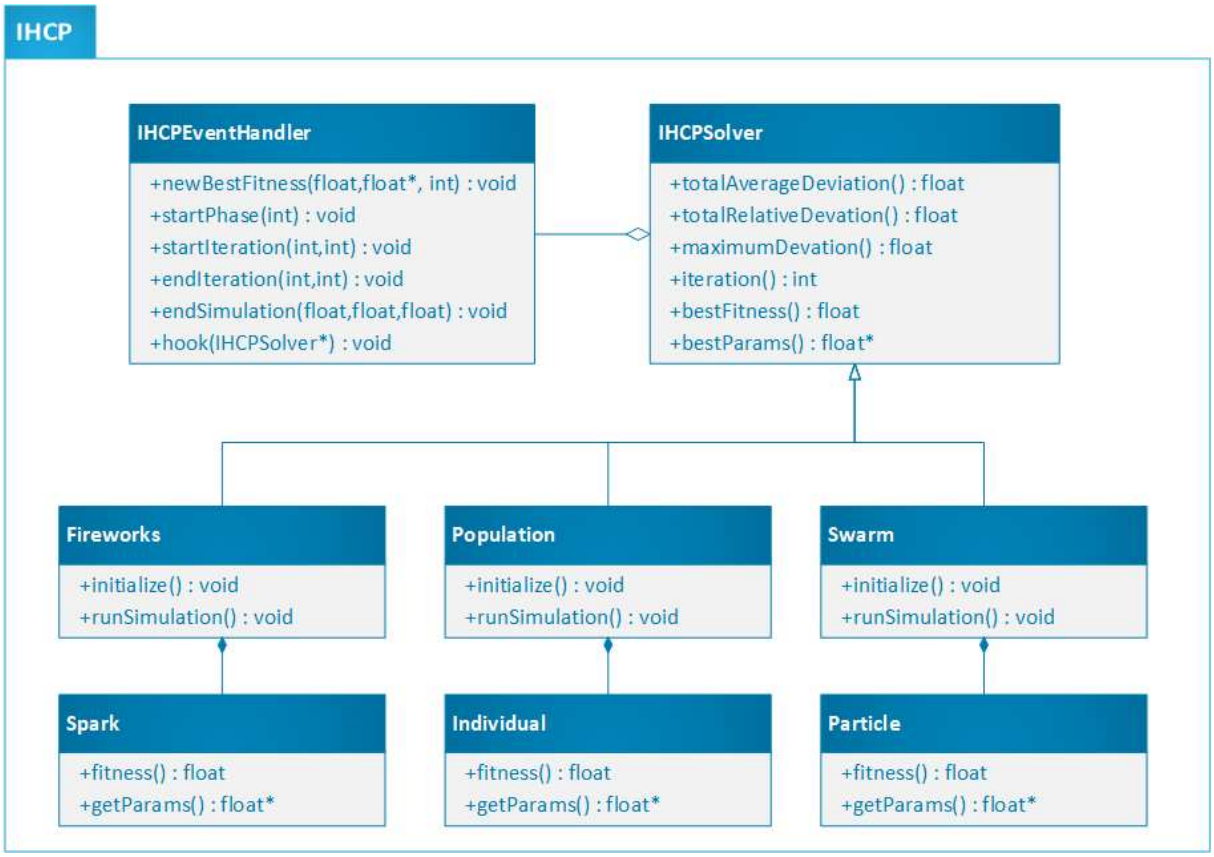
4.3 ábra DHCP (dll) osztálydiagramja

A DHCP (dll) könyvtár olyan osztályokat tartalmaz, amelyek részt vesznek a kétdimenziós hőátadási szimuláció végrehajtásában. A HTCCalculator2D feladata a hőátadási együttható paraméterek függvényekre való leképzése, a diszkrét értékek között elvégzendő interpoláció, paraméterek ellenőrzése és korrekciója. A hőátadás szimulációjának implementációját a HeatConductionCalculator leszármazottjai tartalmazzák. A következő alfejezetben taglalt termelő és fogyasztó modell alapján, ezek a fogyasztók. A DHCPProcessor pedig a termelő, amely ellátja a fogyasztókat munkával.



4.4 ábra DHCP GPU (dll) osztálydiagramja

A DHCP_GPU könyvtár osztályai lényegében ugyanazt valósítják meg, mint a sima DHCP, annyi különbséggel, hogy ezt GPU oldalon. A CUDAHandler végzi a GPU-n való futtatáshoz szükséges kezdeti beállításokat.



4.5 ábra IHCP (dll) osztálydiagramja

A program megvalósítása során fontos szempont volt, hogy modulárisan lehessen kicserélni az optimalizációt elvégző heurisztikákat. Ennek a megvalósításnak az alapja az IHCPSolver ősosztály, amelyből leszármaznak az inverz számításokat végző osztályok. A feladata az, hogy figyelemmel kísérje az iterációs lépéseket és kontrollálja azokat. A könyvtár tartalmazza az Adaptív Tűzijáték algoritmus, Genetikus Algoritmus és a Részecske Raj Optimalizáció implementációját.

Utils (dll) könyvtár segédosztályokat tartalmaz. A Configurations osztály a projekt futtatásához szükséges konfigurációs paramétereket tartalmazó struktúrákat kezeli. A konfigurációs paramétercsoportok az alábbiak:

- Environment
- Processors
- Project
- DHCP
- IHCP
- PSO
- GA
- FWA

FileIOHelper a hőátadás paraméter vektorainak olvasását és írását végzi. A FunctionByValues a 2D függvények tárolásáért felelős. GuidedRandom a megfelelő minőségű (és a hibakeresési célokból szükség esetén megismételhető) véletlenszámgenerálásért felelős. A MainConstants a főbb konstansokat tartalmazza.

Az input mappába kerülnek a DHCP által generált hőmérsékletek ref_[i].txt néven. A konfigurációs fájlban beállított N darab ponton kialakuló hőmérsékletet írja ide az alkalmazás. Az output mappába kerülnek az IHCP eredményei. Amikor az optimalizáció jobb pozíciót talált, akkor az eseménykezelő kiírja ide a htc és a temp függvényeket (pl.: RefTest_AFWA_HTC.txt, RefTest_AFWA_TEMP.txt néven). Az input mappában lévő hőmérsékleteket referenciaként használja fel. A DHCP és az IHCP esetében is szükség van egy fő konfigurációs fájlra. A gyökér mappában vannak eltárolva a referencia hőátadási együtthatók. A HTC és a hőmérsékletek is (idő és érték) formátumban vannak soronként kiírva. A konfigurációs fájl felépítése az 4.2 táblázatban olvasható.

Environment
PseudoRandom
Processors
Project
Output
DHCP
Dimension
InitialTemperature
FinalTemperature
R, L, N, M
DynamicTimePointN
DynamicTimePointRules
EndTime
DeltaT
DeltaTStoreIteration
ZPoints
ReferencePoints
IHCP
DensityIteration
ParameterRules
ReferenceFunction_i
Optimization (pl.: FWA)
Algoritmus paraméterei – pl.: FWA: N, M, NG, A, AM, BM, Lambda
Initialization
StopCondition

4.2 táblázat Konfigurációs paraméterek

4.2. GPU implementáció

4.2.1. Párhuzamosítás szükségessége

Az előzetes tesztek alapján az elkészített rendszer alkalmas az eredetileg kitűzött célokra, tehát képes a hőmérséklet adatok alapján a Hőátadási Együttható értékének becslésére. Komoly hátrányként jelent meg azonban a rendszer erőforrásigénye, ami a heurisztikákat alkalmazó módszerek tipikus problémája.

Ez a probléma az összes raj alapú módszer esetén felmerül, ugyanis ezek mindig azon alapulnak, hogy a populáció minden egyede egy-egy Hőátadási Együttható értéket reprezentál. Ezek fitness értékének kiszámításához pedig meglehetősen költséges számításokat kell

elvégeznünk: le kell futtatnunk egy teljes hőátadási szimulációt az ismert környezeti és az egyed által képviselt paraméterek segítségével, majd pedig ezt össze kell hasonlítanunk a valóságban mért értékekkel.

Egy ilyen szimuláció lefuttatása önmagában még nem tűnik hosszadalmasnak, egy napjainkban átlagosnak tekinthető számítógép esetében kb. 0.1 másodpercet igényel. A heurisztikus módszerek azonban mind iteratív működésen alapulnak, tehát az egyedek időben változó pontokat képviselnek a keresési térben. Emiatt viszont minden iterációban le kell futtatni a szimulációkat a fitness értékek meghatározásához. Így már könnyen belátható, hogy ha egy 1000 elemű populációval futtatok 2000 iterációt, akkor ennek az időigénye már: 1000×2000×0.1másodperc=55 óra. Ez az időszükséglet a gyakorlatban már nagyon erős kompromisszumokat követel meg.

4.2.2. CPU oldali párhuzamosság

A feladat jellege szerencsére lehetővé teszi a megoldás párhuzamosítását. A feladat leginkább erőforrásigényes része a fitness számítás, és ez a részfeladat nagyon eredményesen párhuzamosítható. Az egyes egyedek fitness számítása ugyanis teljesen független egymástól, ami felveti annak a lehetőségét, hogy ezeket ne szekvenciálisan egymást követően, hanem egymással egyidőben, párhuzamosan hajtsuk végre.

A feladat megoldása során egy klasszikus termelő-fogyasztó felépítést valósítottam meg az alábbiak szerint:

- A termelő tulajdonképpen létrehozza a megoldandó feladatokat, ezek jelen esetben az egyes egyedek fitness értékének a kiszámítását jelentik. Tehát minden egyedhez létrehoztam egy külön feladatcsomagot, ami az adott egyed által képviselt Hőátadási Együtthatóhoz tartozó szimuláció lefuttatását és ez így kapott szimulációs eredmények valós mérési eredményekkel való összehasonlítását tartalmazza.
- A fogyasztók pedig végrehajtják ezeket a feladatokat. A termelő által a sorba helyezett feladatokat egyesével kiveszik (a megfelelő zárolási lépéseket végrehajtva a versenyhelyzetek elkerülése érdekében), majd pedig elvégzik a szükséges szimulációkat. Ezt követően egy visszajelzést adnak a kiszámított fitnessről.

Termelő jelen esetben egy van, a heurisztikát futtató processz. Fogyasztó egyidőben több is indítható, célszerűen a rendelkezésre álló processzormagok számának megfelelően. Méréseim alapján elmondható, hogy az így kialakított rendszer tökéletesen beváltotta reményeimet, gyakorlatilag a processzormagok számával lineáris sebességnövekedést sikerült elérni.

4.2.3. GPU oldali párhuzamosság

Mivel a futásidő még így is több óra volt, ezért a grafikus kártyát is szerettem volna felhasználni a további gyorsításhoz. Erre jól használható a CUDA környezet, amelyben általános célú programokat lehet készíteni, amelyek lefuttathatók tetszőleges NVIDIA grafikus kártyán. Ez mindenképpen csábító lehetőség, tudván, hogy a mai videókártyák gyakran több ezer végrehajtóegységgel rendelkeznek szemben a CPU-k 4-8 magjához képest (persze egy GPU végrehajtóegység jelentősen lassabb, mint egy CPU mag, de az összteljesítménybeli különbség még így is kiemelkedő).

A GPU hardver azonban jelentősen különbözik a megszokottól. Ez számos problémát okoz, amelyeket az alábbiak szerint oldottam meg:

- Egy mai grafikus kártyán több ezer végrehajtóegység található, a csúcsteljesítményt csak úgy érhetjük el, ha legalább ennyi szálat futtatunk. → Mivel a szimuláció során nincs szükség több ezer egyedre, ezért az nem tűnt jó megoldásnak, hogy minden szál egy-egy egyed fitness értékét számolja. Ezért a GPU-ra egy teljesen új algoritmust kellett megvalósítanom, ami a szimuláció során használt véges differencia modell által adott rács elemeit eleve párhuzamosan dolgozza fel. Pl. egy 20×40 méretű rács esetén egyidőben 800 szál végzi a szimulációs lépéseket.

- A grafikus kártyán ugyan lehet futtatni több millió szálat, azonban ezeket maximum 1024 szálat tartalmazó, egymástól független blokkokba kell szervezni. → A fenti választással ezt a kihívást is sikerült jól kezelnem. Amennyiben úgy választjuk meg a rácsméretet, hogy annak pontjainak száma 1024 alatt legyen, akkor egy szimuláció lefuttatása megvalósítható egy GPU blokkon belül. Ezzel persze még nem sikerült kihasználni a GPU képességeit, azonban vegyük figyelembe hogy ezt a szimulációt nem csak egyszer, hanem a teljes populáció minden egyedére le kell futtatni. Ennek megfelelően egy 1000 elemű populáció esetében, 20×40 méretű rács esetén összesen

800 000 szálat tudtam indítani, amivel már valóban hatékonyan ki lehet használni a GPU minden erőforrását.

- Grafikus kártyák még nem rendelkeznek olyan fejlett automatikus cache kezeléssel, mint a CPU-k. → Emiatt úgy kellett módosítanom az algoritmust, hogy az minél kevesebb eszközmemória hozzáféréssel bírjon. A grafikus kártyák rendelkeznek úgynevezett onchip shared memóriával, ami nagyon gyors, viszont meglehetősen korlátos méretű (kb. 48Kb). Sikerült úgy optimalizálnom a szimulációt végző algoritmust, hogy maga a rács, illetve a szükséges állandó adatok elférjenek a shared memóriában, a további nem változó adatok (pl. a valós mérési eredmények) pedig a még szintén gyors elérést biztosító konstans memóriába kerüljenek.

- A grafikus kártya adatpárhuzamos módon 32 darab szálat egy warpként összekapcsolva futtatja a programokat → GPU fejlesztés során nehezen megoldható problémát jelent a warp divergencia fogalma. Ez akkor merül fel, ha az egymás mellett futó szálak nem pontosan ugyanazt a kódot futtatják, ilyenkor esetenként várniuk kell egymásra. Egy újszerű ütemezés kidolgozásával azonban sikerült jelentősen csökkentenem ennek a divergenciának a mértékét.

4.2.4. Hibrid megvalósítás

A GPU megvalósítás jelentős sebességnövekedéssel járt, bizonyos esetekben akár hétszeres gyorsulást is el tudtam vele érni. Ez persze jelentősen függ a konkrét feladattól: amennyiben a párhuzamosan végrehajtható feladatok száma alacsony (pl. csak egy egyed van a populációban) akkor még a CPU volt gyorsabb, 10 darab egyednél már a GPU, több száz egyednél pedig már egyértelmű volt a GPU előnye.

Fontos azonban megjegyezni, hogy a két implementáció nem zárja ki egymást. Mivel az egyes fitness számítások egymástól teljesen függetlenek, ezért nincs akadálya, hogy a populáció egy részét a CPU, másik részét pedig a GPU dolgozza fel. Ezért kidolgoztam a párhuzamosság egy még magasabb szintjét, amikor már nem is csak egy, hanem akár több GPU is dolgozhat párhuzamosan a CPU magok mellett. Itt a terhelés elosztás már meglehetősen összetett, ezért kidolgoztam egy saját módszertant, amivel ezt hatékonyan el lehet végezni.

Az első példában említett 1000 elemű populáció esetében egy 4 magos CPU-t és két GPU-t használva az optimális kiosztás az alábbi: 90 elemet kezel a CPU, 455 elemet az első GPU, újabb 455 elemet pedig a második GPU. Ennek eredményeként akár 40x-es sebességnövekedést is el lehet érni. Ezzel a több napos szimulációk is lefuttathatók akár 1 óra alatt is.

4.2.5. CPU és GPU oldali megvalósítások futási idejének összehasonlítása

A 4.1 táblázatban a CPU és GPU oldali megvalósítás eredményeit jelenítem meg. Ezek nem hibrid megoldások, vagyis mindkét megvalósítás csak azonos típusú fogyasztókat tartalmaz. A populáció 100 egyedből áll. A teszteket addig futtattam, amíg 2 227 662 fitness számítási lépést nem végeztek el. A rendszer optimalizációs komponensének az Adaptív Tűzijáték Algoritmust választottam. A teszteket futtató környezet: Intel Core i5 3,3GHz, NVDIA GeForce GTX 1060 6GB grafikus kártya, 16 GB DDR3 RAM. A CPU megvalósítás közel 15,24x több idő alatt végzett, mint a GPU megvalósítás. Ezért a következő fejezetekben taglalt tesztjeim során is mindig a GPU oldali és Hibrid megvalósításokat alkalmaztam.

	CPU	**GPU**	
	Futási idő (óra)	**Futási idő (óra)**	**Fitness számítás (db)**
100 darab egyed	20,42	1,34	2 227 662

4.1 táblázat CPU és GPU oldali megvalósítás eredményei

5. Eredmények értékelése

5.1. A vizsgálat módszertana

A kifejlesztett becslési módszer vizsgálata során arra kerestem a választ, hogy egyrészt az inverz hőátadási probléma megoldására hivatott számítási eljárás alkalmas-e a Hőátadási Együtthatók megfelelő pontosságú rekonstrukciójára. Másrészt, az Adaptív Tűzijáték Algoritmus becslési hatékonyságát vizsgáltam oly módon, hogy egy adott inverz hőátadási feladatot az AFWA módszer mellett egy másik, népszerű optimalizálási módszerrel is megoldottam és az eredmények pontosságát, illetve a becsléshez igénybe vett számítási lépések számát hasonlítottam össze. Harmadrészt, egy hengeres geometriájú próbatest repceolajban való hűtése során rögzített hőmérsékleti görbéket rekonstruáltam és számítottam ki a henger felületeire jellemző, helykoordináta- és időfüggő Hőátadási Együtthatókat a becslési módszer segítségével. Ez utóbbi vizsgálattal a kidolgozott számítási eljárás gyakorlatban való alkalmazhatóságát kívántam demonstrálni.

Az időben változó hőátadás közben kialakuló Hőátadási Együttható predikciójára kifejlesztett eljárást az alábbi módszertan alapján vizsgáltam:

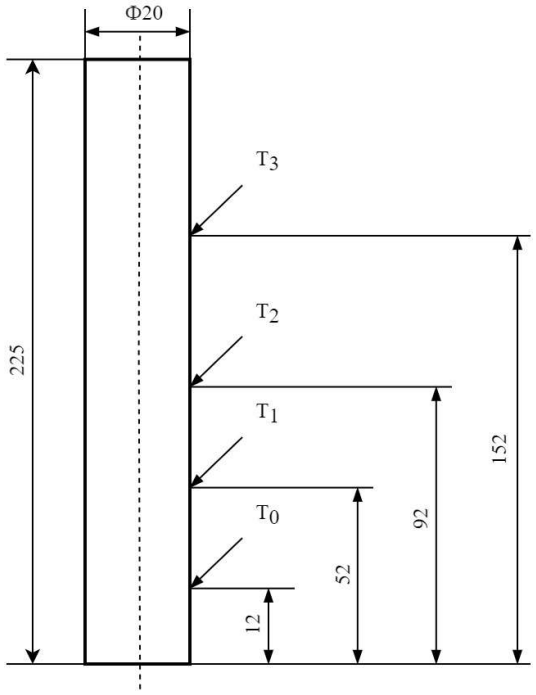
1. Hipotetikus hőátadási együttható függvényeket $h_1^m(t), h_2^m(t) \dots h_n^m(t)$ állítottam elő, melyek a hőátadás változását az idő függvényében jellemzik a henger véglapjától mért különböző távolságokban $(1{,}2 \dots n)$.
2. Lehűlési szimulációkat végeztem a hipotetikus hőátadási együtthatók alkalmazásával, a $h_1^m(t), h_2^m(t) \dots h_n^m(t)$ függvényeket egy henger palástfelületéhez tartozó termikus peremfeltételében vettem figyelembe. A hőátadási együttható komplex (idő és helykoordináta szerinti) függvényét a $h_1^m(t), h_2^m(t) \dots h_n^m(t)$ függvényeknek és a henger alapjától mért távolságoknak bilineáris interpolációjával határoztam meg. A szimulációk eredményeként a henger alkotójától 1 mm mélységben kialakult lehűlési görbéket $T_0^m(t), T_1^m(t), T_2^m(t), T_3^m(t)$ rögzítettem.
3. A kidolgozott predikciós módszer segítségével számítottam a Hőátadási Együttható függvényeket $[h_1^c(t), h_2^c(t) \dots h_n^c(t)]$. A számításaimhoz az előző lépésben származtatott hőmérsékleti görbét, mint „mérésből származó” hőmérsékleti mintát $T_0^m(t), T_1^m(t), T_2^m(t), T_3^m(t)$ alkalmaztam.

4. A hipotetikus $[h_i^m(t)]$ és a rekonstruált $[h_i^c(t)]$ Hőátadási Együttható függvények összehasonlítása alapján elemeztem a kidolgozott becslési eljárást.

A numerikus kísérletekhez a dolgozatom 3. fejezetében bemutatott hőátadási modellt alkalmaztam az 5.1 táblázat és az 5.1 ábrán lévő paraméterek felhasználásával.

A szimuláció paraméterei	
Henger átmérője	20 mm
Henger hossza	225 mm
Munkadarab kezdeti hőmérséklete	850 °C
Hűtési idő	170 sec

5.1 táblázat A szimulációs paraméterek



5.1 ábra Mérési pontok (termoelemek) elhelyezkedése a munkadarabon (méretek mm-ben)

A hőátadási együttható predikciójának minősítéséhez az alábbi származtatott paramétereket vizsgáltam:

- A legnagyobb hőmérséklet különbség, T_{diff}[°C]: a számított és mért hőmérsékleti görbék bármely időpontjában kialakult legnagyobb eltérés.

- A becslés pontosságának minősítéséhez a teljes átlagos eltérés (Total Average Deviation, E_{Fa}) és a teljes relatív eltérés (Total Relative Deviation, E_{Fr}) metrikákat használtam, amelyek a

$$E_{Fa} = \sqrt{\sum_{i=1}^{4} \frac{1}{t_{max}} \left(\sum_{t=0}^{t_{max}} \left[T_i^c(t) - T_i^m(t) \right]^2 \right)} \quad (5.1)$$

és

$$E_{Fr} = \sqrt{\sum_{i=1}^{4} \frac{1}{t_{max}} \left(\sum_{t=0}^{t_{max}} \left[\frac{T_i^c(t) - T_i^m(t)}{T_i^m(t)} \right]^2 \right)} \quad (5.2)$$

formulákkal definiáltak.

- A Hőátadási Együttható becsléséhez szükséges számítási lépések száma, $N_{pred}[db]$: a célfüggvény értékére vonatkozó $(S < 200) \wedge (T_{diff} < 10\ ℃) \wedge (E_{fa} < 2.5\ ℃) \wedge (E_{fr} < 0.03\ ℃)$ feltétel teljesítésig végzett számítási lépések száma.

A legnagyobb hőmérséklet különbség, $T_{diff}[℃]$ értéke a szakirodalomban fellelhető források szerint 10 ℃, így a vizsgálataimnál ezt az értéket tekintem az elfogadási határértéknek.

5.2. A becslési eljárás alkalmazhatóságának vizsgálata

Az AFWA módszer teszteléséhez az 5.1 táblázatban feltüntetett paramétereket vettem figyelembe. Numerikus vizsgálataimat konstans és időben változó Hőátadási Együttható feltételezése mellett végeztem.

Adaptív Tűzijáték Algoritmus paraméterei	
$\boldsymbol{n}$	**10**
$\boldsymbol{m}$	**200**
$\boldsymbol{\hat{m}}$	**5**
$\boldsymbol{A}$	**100**
$\boldsymbol{am}$	**2**
$\boldsymbol{bm}$	**100**
$\boldsymbol{\lambda}$	**1,3**

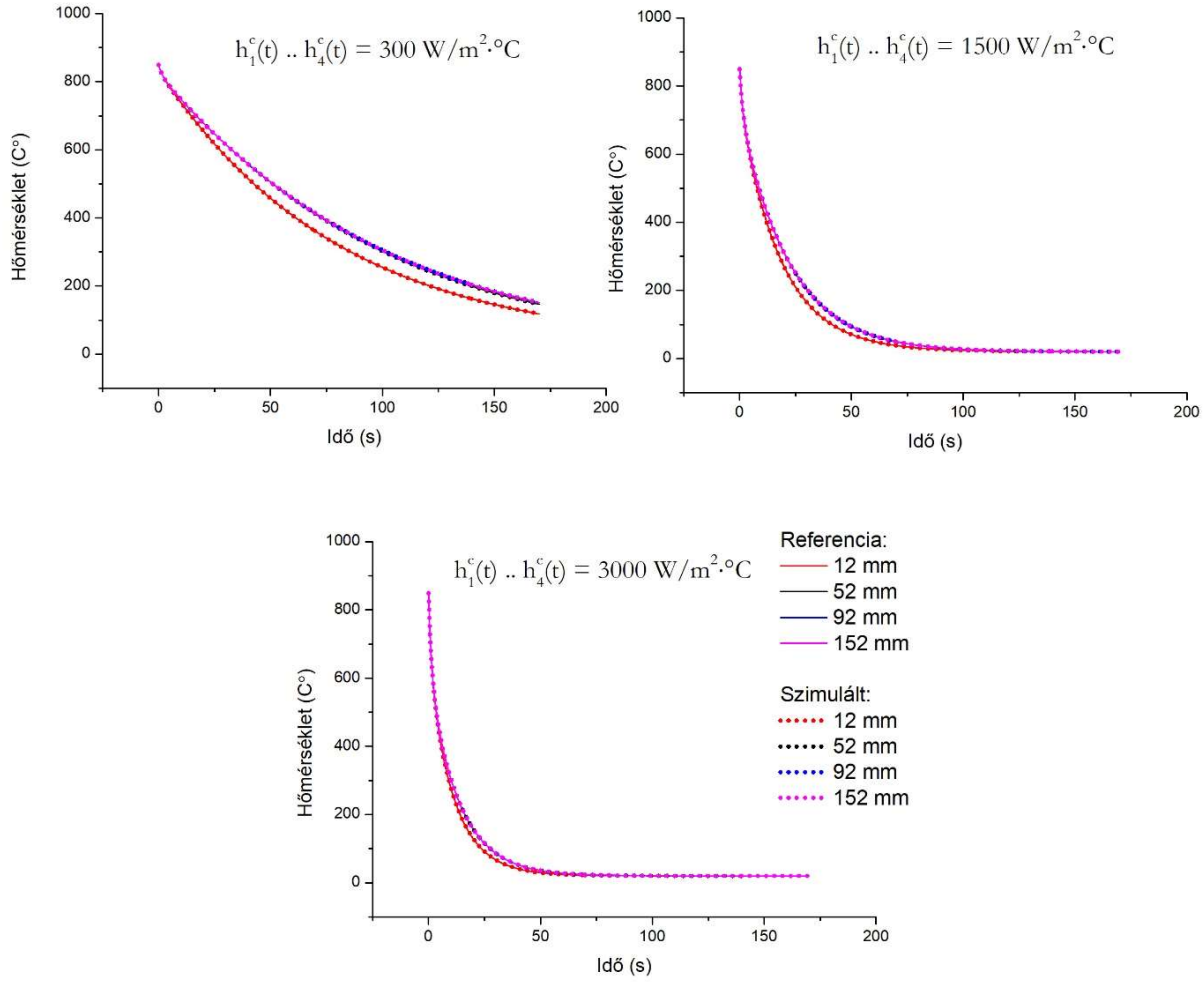
5.1 táblázat AFWA paraméterei

5.2.1. Konstans Hőátadási Együttható függvények rekonstrukciója

Háromféle $h_i^m(t)$ függvényt definiáltam a konstans Hőátadási Együttható mellett végzett tesztekhez [$h_1^m(t)$=300$\frac{W}{(m^2 \cdot ℃)}$, $h_2^m(t)$=1500$\frac{W}{(m^2 \cdot ℃)}$ és $h_3^m(t)$ (t)=3000$\frac{W}{(m^2 \cdot ℃)}$], melyeknél azt

feltételezzük, hogy a teljes 0-170 másodperces időperiódusban konstans a Hőátadási Együttható értéke.

A számítások eredményei, azaz a henger palástja alatti 1 mm-es mélységben, négy ponton kialakult hőmérséklet időbeli változását a 5.2 ábrán látható görbék mutatják. A teszteléshez számított metrikák értékei a 5.2. táblázatban láthatóak. A tesztek eredményei arra mutatnak rá, hogy a célfüggvény kiértékelés után mindegyik Hőátadási Együttható alkalmazása mellett a mért és számított hőmérséklet ciklusok különbsége igen csekély mértékű [0.93℃; 0.76℃; 0.46℃]. Ez a különbség a szakirodalmi források szerint elfogadható, és ezek szerint megállapítható, hogy az AFWA algoritmus alkalmas konstans Hőátadási Együtthatók megbízható predikciójára.



5.2. ábra – Lehűlési görbék konstans $h_i^c(t)$ mellett

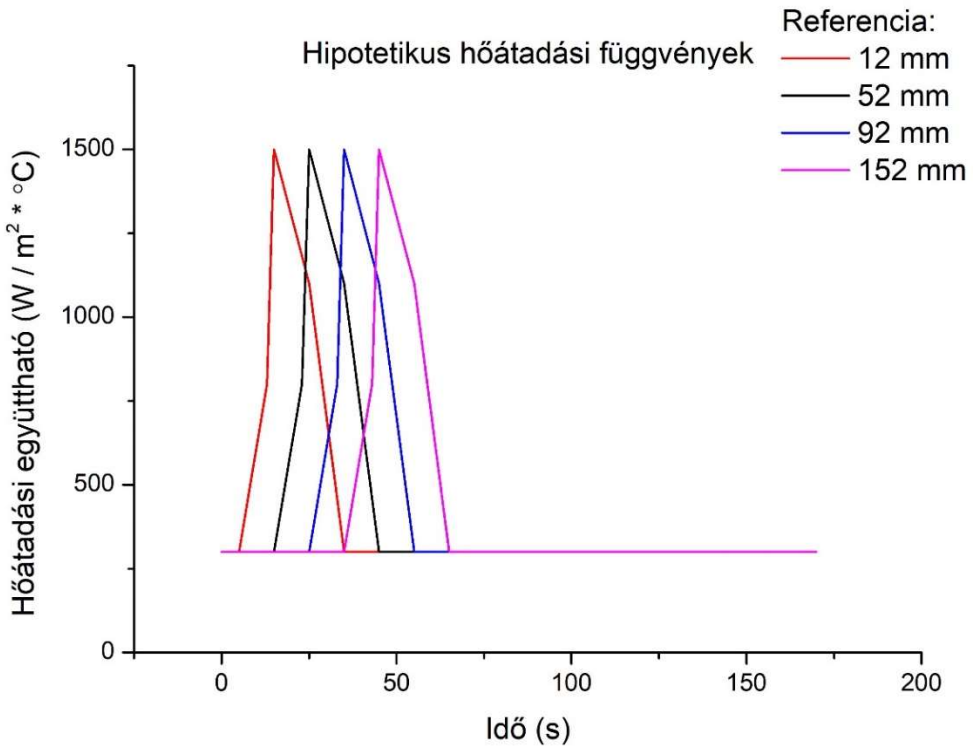
Konstansok	T_{diff}[°C]	S	E_{Fa}	E_{Fr}
$h_1^c(t)=300\frac{W}{(m^2\cdot°C)}$	0.93	12.39	0.15	0.0007
$h_2^c(t)=1500\frac{W}{(m^2\cdot°C)}$	0.76	15.68	0.19	0.0016
$h_3^c(t)=3000\frac{W}{(m^2\cdot°C)}$	0.46	8.86	0.1075	0.0017

5.2. táblázat A konstans Hőátadási Együtthatók mellett végzett tesztek eredményei

5.2.2. Időben változó Hőátadási Együttható függvények rekonstrukciója

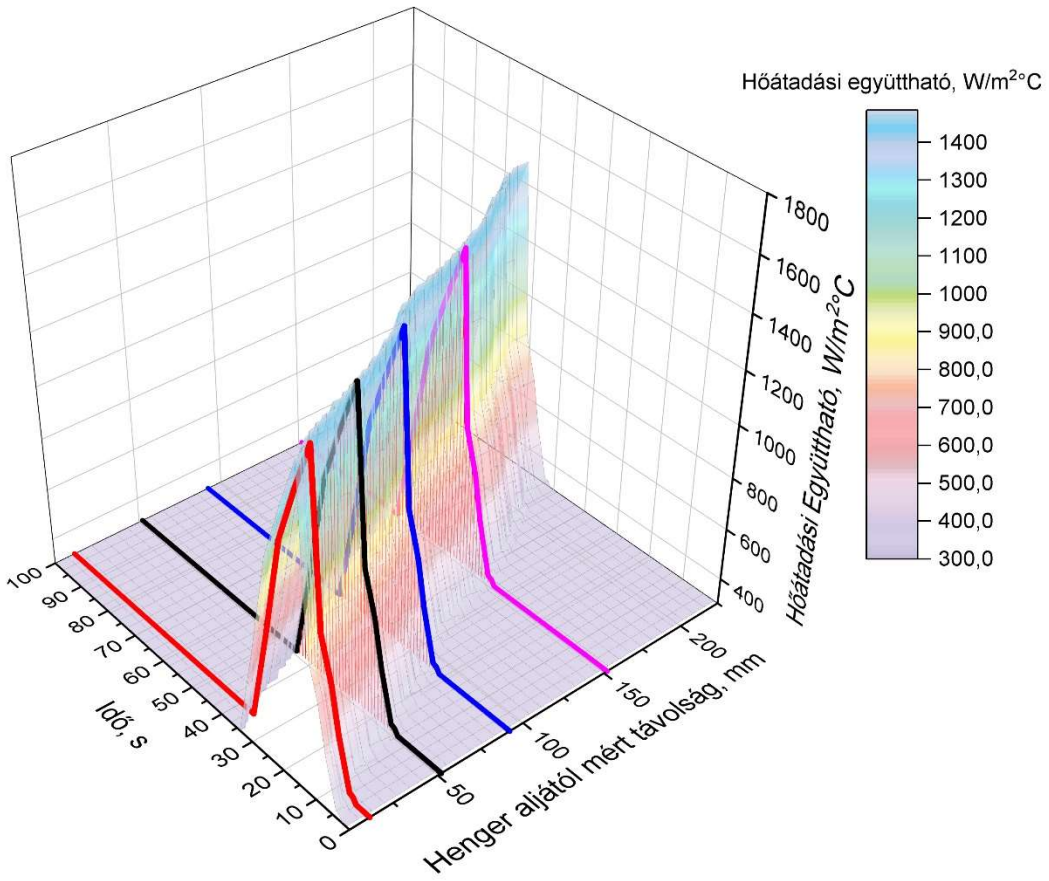
Az instacioner, azaz időben változó hőátadás közben kialakuló Hőátadási Együttható függvények számítását és a számítási teljesítmény vizsgálatát az alábbi megfontolások szerint végeztem.

A numerikus modellben definiált termoelem helyekhez rendelhető, az idő függvényében változó hipotetikus Hőátadási Együttható függvényeket $[h_i^m(t)]$ hoztam létre (5.3a ábra). A henger felületeihez tartozó komplex Hőátadási Együttható függvényt az $[h_i^m(t)]$ függvények és a termoelemhez tartozó helykoordináták bilineáris interpolációja alapján állítottam elő (5.3b. ábra).

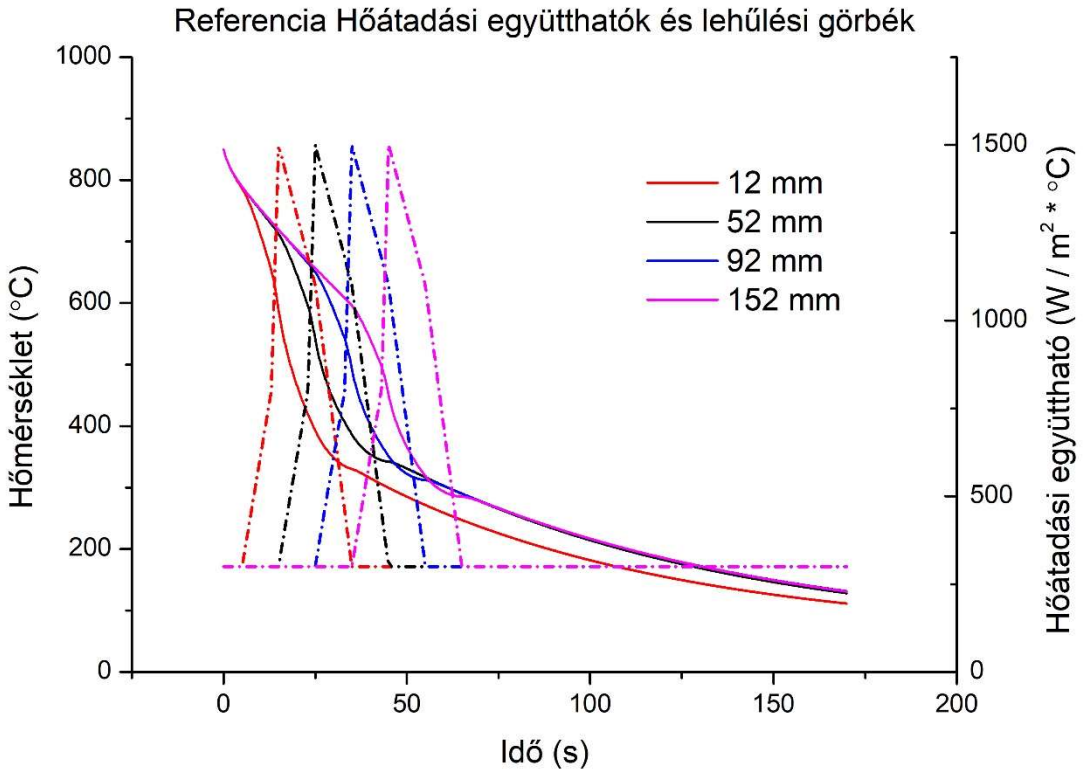


5.3a. ábra Teszthez felhasznált hengeres próbatest 4 különböző pontjához tartozó hipotetikus Hőátadási Együtthatók az idő függvényében (12=piros, 52=fekete, 92=kék, 152=magenta)

A Hőátadási Együttható függvényeket polinomokkal írtam le. A tesztelés céljából generált polinomok fokszáma 5. Az egyes helykoordinátákhoz rendelt $h_i^m(t)$ függvények azonos karakterisztikájúak, a hőátadás értéke minimum 300 $\frac{W}{(m^2 \cdot °C)}$ és maximum 1500 $\frac{W}{(m^2 \cdot °C)}$ érték között változik.

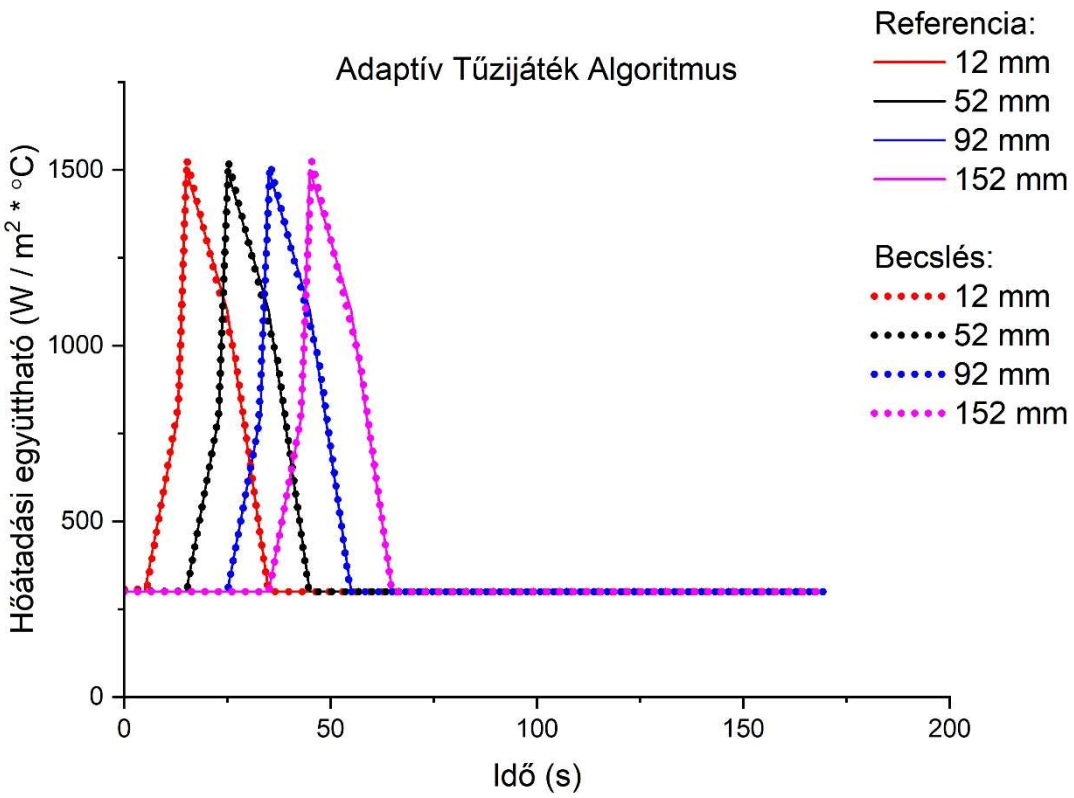


5.3a. ábra Teszthez felhasznált, a hengeres próbatest felületeihez tartozó hipotetikus Hőátadási Együtthatók a helykoordináta és az időfüggvényében.

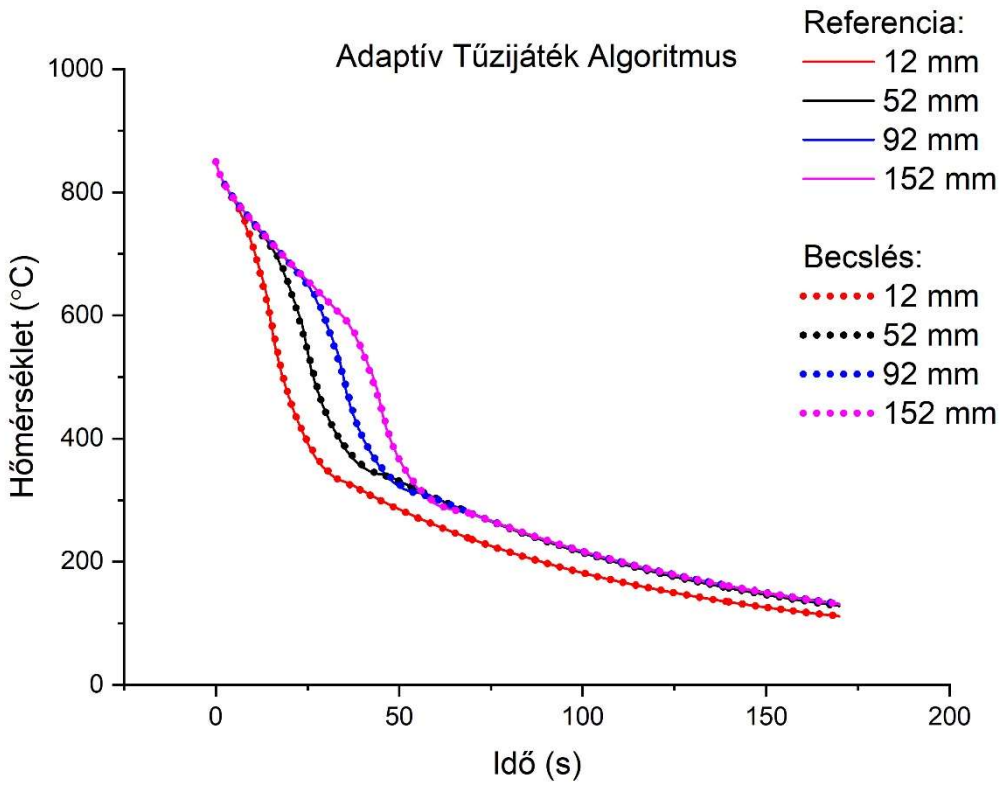


5.4. ábra Jelentős hőátadás a hűtés első 70 másodpercén belül

Az AFWA módszerrel végeztem el a függvények becslését. A teszt eredményeit, azaz a Hőátadási Együtthatókat és a lehűlési görbéket a 5.5. és 5.6. ábrák szemléltetik. A 5.3. táblázatban foglaltam össze a számítás megfelelőségét alátámasztó metrikákat.



5.5 ábra AFWA által rekonstruált hőátadási függvények

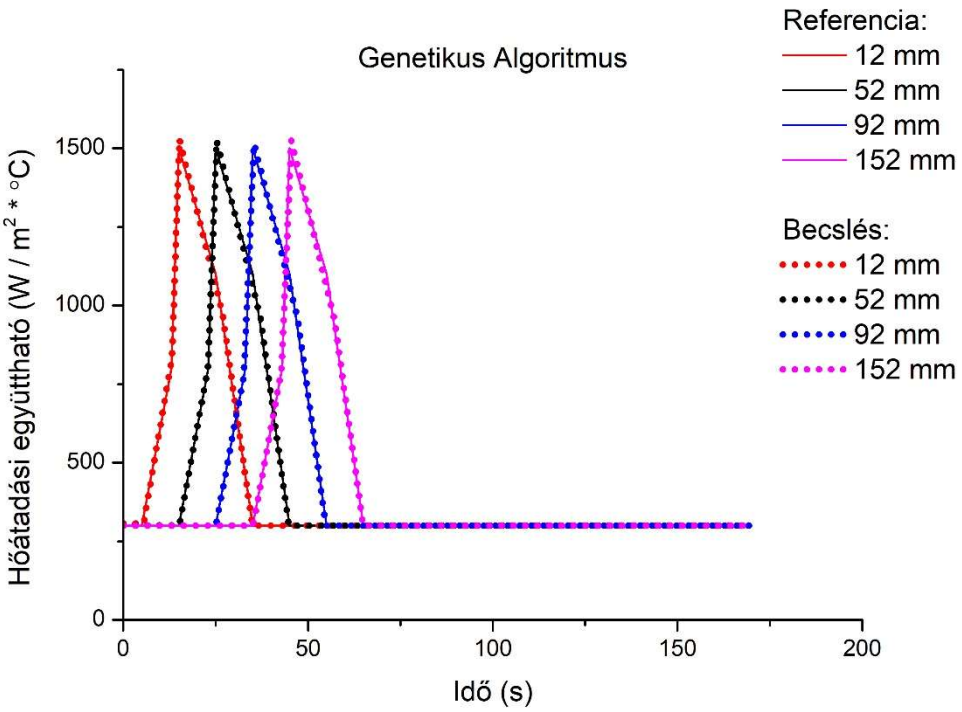


5.6. ábra AFWA által rekonstruált lehűlési görbék

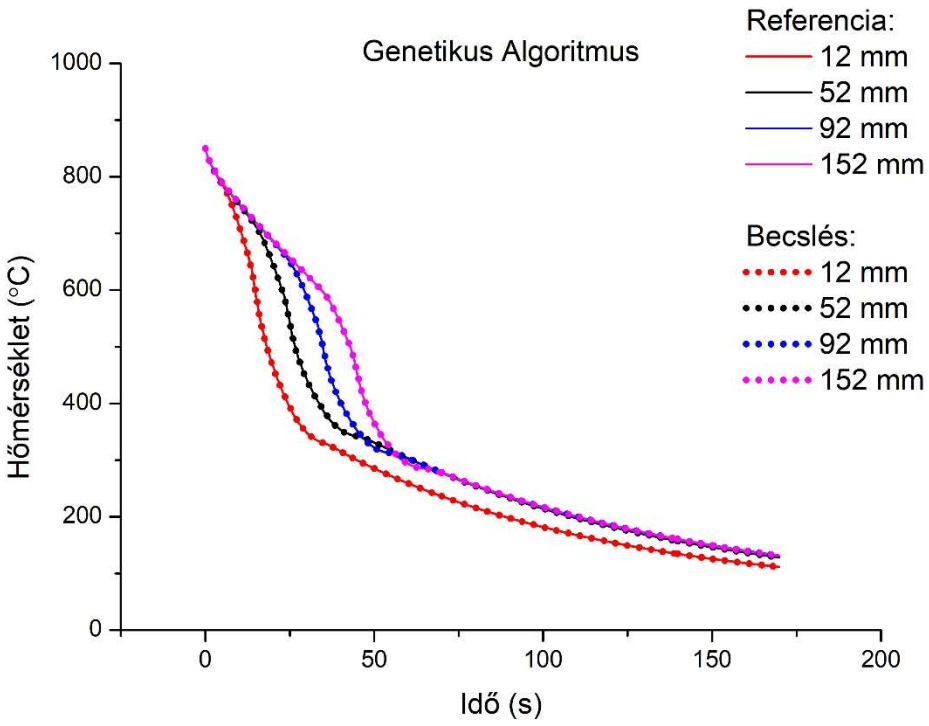
Hipotetikus	T_{diff}[°C]	S	E_{Fa}	E_{Fr}
Adaptív Tűzijáték Algoritmus	5.11	69.54	0.8430	0.0020

5.3. táblázat AFWA által elért eredmény

Vizsgálataimat kiterjesztettem egy másik populáció alapú optimalizálási eljárás alkalmazására, majd összevetettem az AFWA módszerrel. Nevezetesen a rulettkerék szelekción alapuló Genetikus Algoritmus (GA) számítási hatékonyságát vizsgáltam ugyanannak az inverz hővezetési feladatnak a megoldására. A tesztjeim során ugyanazon hipotetikus $h_i^m(t)$ függvények által előállított $T^m(t)$ hőmérsékleti mintát vettem bemeneti adatnak a másik optimalizálási eljárással megvalósított Hőátadási Együttható becsléshez. A Genetikus Algoritmus segítségével előállított hőátadási függvényeket és lehűlési görbék az 5.7 és 5.8 ábrákon láthatók.



5.7. ábra GA által rekonstruált hőátadási függvények



5.8. ábra GA által rekonstruált lehűlési görbék

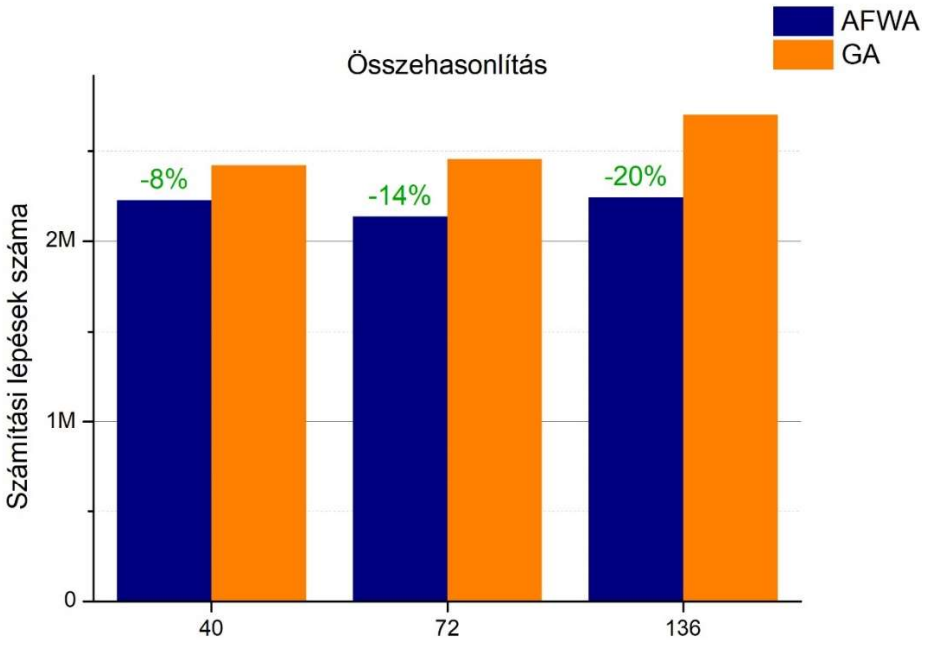
A kimunkált numerikus eljárás segítségével rekonstruáltam Hőátadási Együttható függvényeket, olyan módon, hogy az egyes termoelemek által meghatározott helykoordinátákhoz rendelt, időben változó Hőátadási Együttható függvényeket $[h_i^m(t)]$ különböző fokszámú polinomokkal írtam le. A különböző fokszám egyrészt a becslés pontosságát befolyásolja, hiszen alacsony fokszám mellett a komplex topológiával leírható függvények becslése csak bizonyos határok mellett teljesíthetők. Másrészt – mivel a polinomok fokszámának összege (a négy polinom fokszámainak összege) valójában megegyezik az optimalizálási problématér dimenziójának számával – a fokszám emelkedésével a számítási idő is növekedett.

A különböző polinom fokszámú becslések mellett végzett számítások alapján a becslési eljárás hatékonyságát jellemeztem kvantitatív módon. Az 5.9 ábra vízszintes tengelyén az ismeretlen paraméterek száma szerepel, a függőleges tengelyen a paraméterek rekonstruálásához szükséges számítási lépéseknek a száma látható. A számításokat mindkét módszerrel (FWA és GA) az ábrán feltüntetett leállási feltételig végeztem, a lépésszámokat ugyancsak a diagramon jelenítettem meg. Összesen 2-2-2 db teszt eredménye látható az 5.9 ábrán az AFWA és GA módszerekkel végzett számításokról.

Az első tesztnek az eredményeit az első két oszlop mutatja az 5.9. ábrán. Mivel a hőátadási függvényeket 5 fokszámú polinommal írtam le, ezért 2×5×4=40 adatot kellett rekonstruálnom az optimalizáció alkalmazásával. Ebben az esetben az AFWA módszer 8%-al kevesebb számítási lépés alatt tudta elérni a leállási feltételt, mint a GA módszer.

A Hőátadási Együttható függvények 9 fokszámú polinommal történő közelítése esetében 72 paraméter rekonstruálását végeztem a két optimalizálási algoritmussal. Az AFWA módszerrel 14 %-al kevesebb lépés elegendő volt a leállási feltételek eléréséhez.

Végül 17 számpárral (17 fokszámú polinom) definiált függvények esetében (az optimalizálási problématér dimenziószáma 136), az AFWA optimalizálási módszer 20%-al kevesebbszer hívta meg a fitness függvényt. Mivel mind a három paraméterszámú optimalizációnál az Adaptív Tűzijáték Algoritmus kevesebb fitness függvény kiértékelést igényelt a Genetikus Algoritmushoz képest, ezért megállapítható, hogy az AFWA optimalizálási eljárást célszerű alkalmazni ezekhez a feladatokhoz.



	Paraméterek száma		
	40	72	136
AFWA	2 227 662	2 137 080	2 243 833
GA	2 419 890	2 457 435	2 701 980

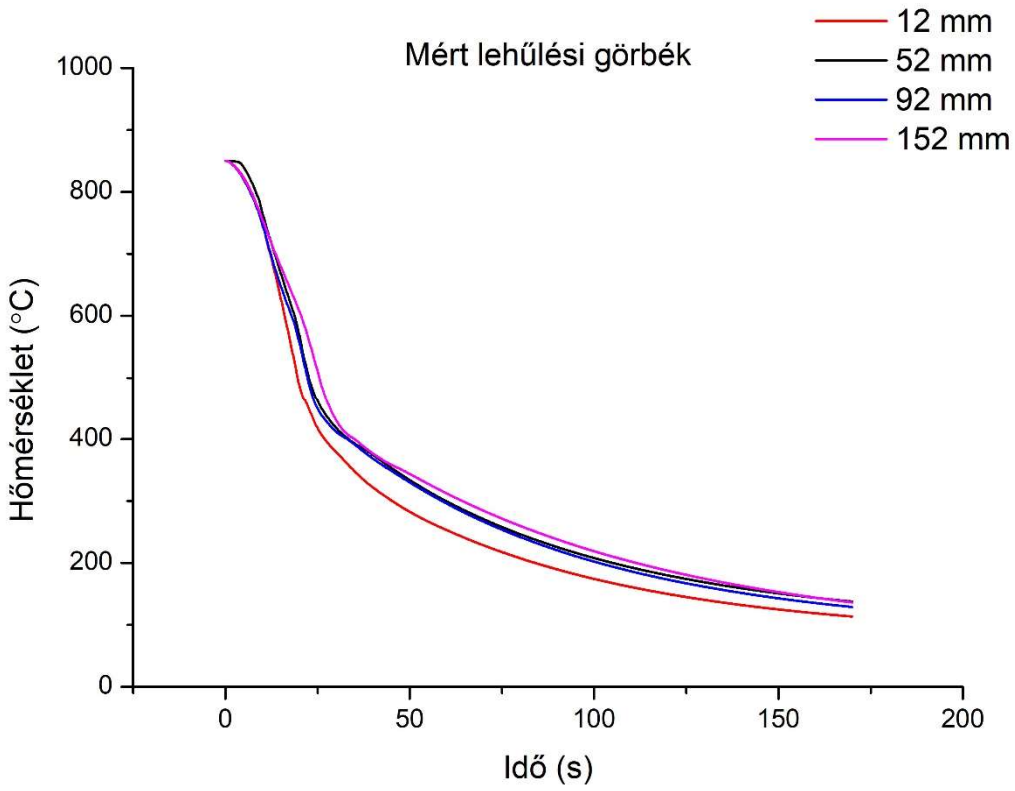
5.9. ábra AFWA és GA számítási lépéseinek összehasonlítása

5.2.3. Hőátadási Együttható becslése valós edzési kísérletek mérései alapján

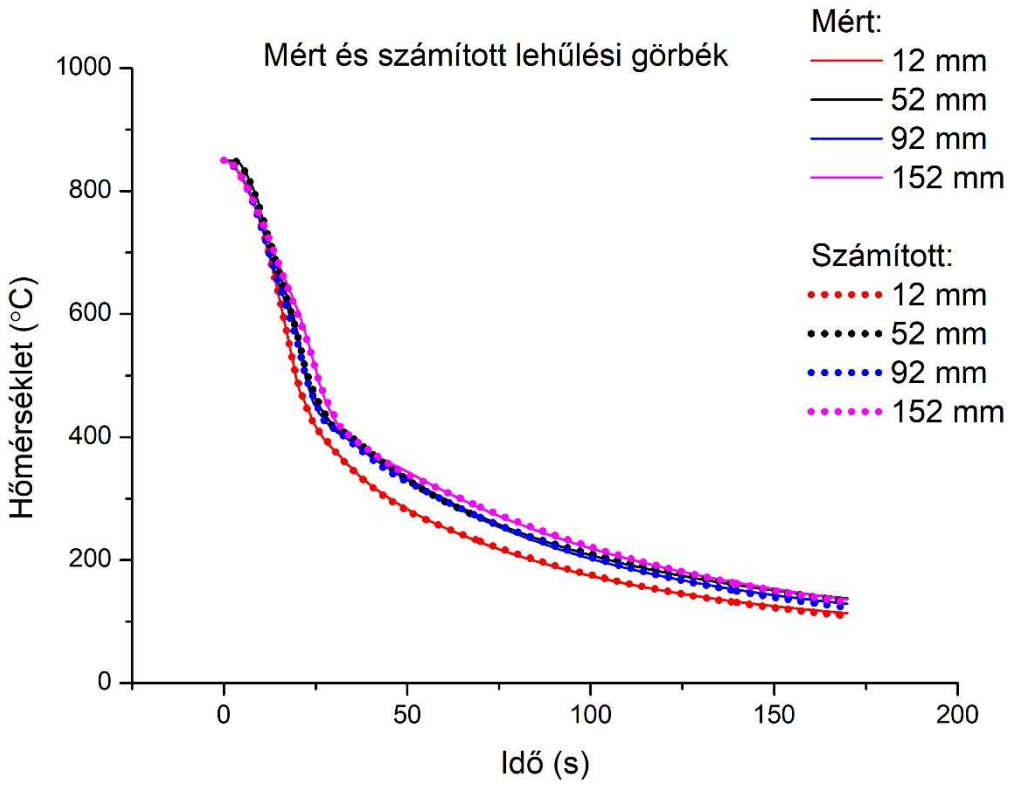
Az 5.1 ábrán látható próbatest alkalmazásával bemerítéses edzési kísérleteket hajtottunk végre organikus hűtőközegben. [16] A hengeres próbatestet 850 °C kezdő hőmérsékletről helyeztük bele álló (vertikális pozícióban) 30 °C -os repceolajba és a kísérlet során rögzítettük az előző fejezetben bemutatott próbatestben azonos helyekre telepített K-típusú termoelemek jelét. A lehűlési görbéket a 5.10 ábra illusztrálja, melyen jól látható, hogy az azonos időpillanatban az egyes helyeken mért hőmérsékletek különbözőek. Ez a hőmérsékleti inhomogentiás azzal magyarázható, hogy az organikus közegben a hőelvonás kinetikája a vertikális helyzetben lévő henger felületén eltérő módon megy végbe utalva a 2. fejezetben bemutatott komplex hőátadási jelenségre.

Az így kapott lehűlési görbékkel és a kimunkált inverz becslési módszer segítségével állítottuk elő a Hőátadási Együttható függvényeket. Az inverz számítások eredményeként előálló lehűlési

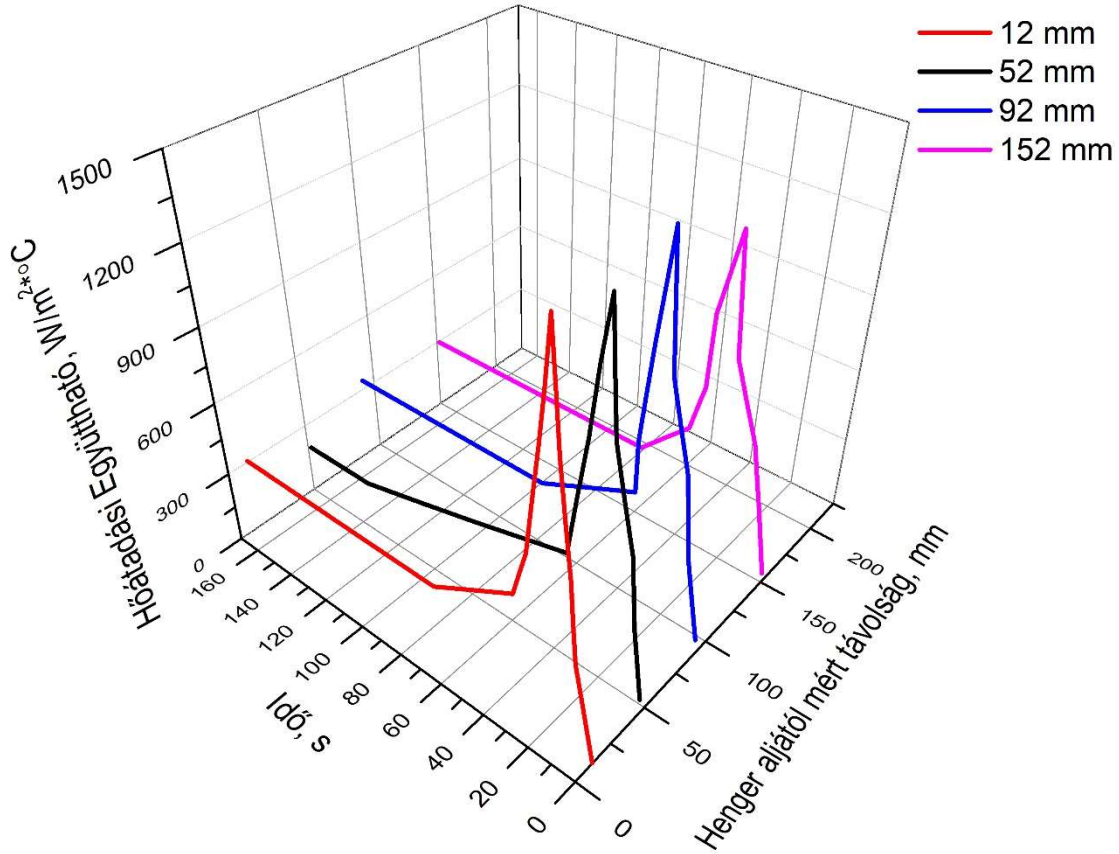
görbéket és a mért hőmérsékleti jeleket az 5.11 ábra szemlélteti. Amint az ábrán látható, az azonos helyeken mért és számított görbék közötti különbség csekély. Az 5.12. ábrán a becsült Hőátadási Együtthatók láthatók.



5.10. ábra Mért lehűlési görbék



5.11. ábra Mért görbék becslése az inverz számítások alkalmazásával



5.12 ábra Becsült Hőátadási együtthatók

Repceolaj	T_{diff}[°C]	S	E_{Fa}	E_{Fr}
Adaptív Tűzijáték Algoritmus	8.14	206	2.49	0.0129

5.4. táblázat Eredmények

Az AFWA alkalmazásával elvégzett inverz számítások eredményei az 5.4. táblázatban láthatók. A legnagyobb hőmérsékleti különbség kisebb, mint szakirodalomban elfogadott tolerancia határként megszabott 10°C. A szimuláció 2:04:13 órán keresztül futott. Az eredmények azt mutatják, hogy az eljárás képes a rögzített hőátadási folyamat megfelelő pontosságú becslésére.

6. Összefoglalás és következtetések

Kutatásom célja olyan számítási eljárás kidolgozása volt, mely alkalmas a gyors hőmérsékletváltozással járó hőátadási viszonyok között kialakuló, az időben és a helykoordináta függvényében változó Hőátadási Együtthatók becslésére.

A célkitűzés teljesítése egy Inverz Hőátadási Probléma megoldását feltételezte. Matematikai modellt és számítási módszert dolgoztam ki az Inverz hőátadási probléma (IHCP) megoldására, melyben központi elemként az újdonságnak tekinthető raj alapú, robusztus Adaptív Tűzijáték Algoritmus (AFWA) optimalizálási eljárást használtam fel. A matematikai módszert keretrendszerbe implementáltam, mely a kétdimenziós tengelyszimmetrikus hőátadási modell alapján egy hengeres test felületén kialakuló Hőátadási Együttható becslését teszi lehetővé.

A rendszer grafikus gyorsítókra való implementálásával többszörös sebességnövekedést értem el. Ez utóbbi fejlesztési lépés több mint húszszoros számítási sebességnövekedést jelentett, ami lehetővé teszi összetett valós problémák megoldását is elfogadható időn belül. Az elkészült algoritmus képes egyidőben az összes CPU mag és akár több GPU használatára is.

A kidolgozott numerikus eljárás alkalmazhatóságának vizsgálatához módszertant dolgoztam ki, mely az eljárás pontosságának ellenőrzéséhez első lépésben a hipotetikus Hőátadási Együtthatók alkalmazásával generált hőmérséklet mintákat használja bemeneti adatként. A becslési eljárás a második lépésben „rekonstruálja” a Hőátadási Együttható függvényt. A harmadik lépésben a hipotetikus eredeti és a numerikusan becsült Hőátadási Együttható függvények összehasonlítása alapján minősíthető a predikció megfelelősége. Ennek a módszertannak a segítségével vizsgáltam az AFWA módszer alkalmazásával működő becslési eljárást. Az időben és térben változó Hőátadási Együttható becslésénél a tesztelési eredmények a predikció pontosságát igazolták.

Az AFWA számítási hatékonyságát a Genetikus Algoritmussal hasonlítottam össze. A numerikus vizsgálatok alapján megállapítható, hogy az AFWA algoritmussal a Hőátadási Együtthatók kevesebb számítási lépéssel becsülhetők, mint a GA optimalizálási eljárással.

Eredményeimről konferenciacikket nyújtottam be az IEEE SACI2021 nemzetközi konferenciára.

Köszönetnyilvánítás

Köszönetet szeretnék mondani konzulenseimnek Dr. Felde Imrének és Dr. Szénási Sándornak a dolgozatomban való segítségért.

Szeretném megköszönni az Óbudai Egyetem GPGPU Programozás Kutatócsoportjának a támogatását.

Ezen kívül köszöntetet szeretnék mondani az Új Széchenyi Terv keretein belül az EFOP-3.6.1-16- 2016-00010 azonosító számú projekt támogatásáért.

Irodalomjegyzék

[1] B. S. H. M. E. M. A. Majorek, „Influence of heat transfer on development of residual stresses in quenched steel cylinders,” *Steel research,* %1. kötet4, pp. 146-151, 1994.

[2] G. Leidenfrost, „De Aqua Communis Nonnullis Qualitatibus Tractatus 1756.,” in *Int. J. Heat Mass Transfer 9*, 1966, p. 1153–1166.

[3] Quenching Theory and Technology doi:10.1201/9781420009163, CRC Press, 2010.

[4] H. R. B. O. M. N. Ozisik, „Inverse Heat Transfer Fundamentals and Application,” 2000.

[5] O. M. Alifanov, „Inverse Heat Transfer Problems,” Springer-Verlag, Berlin, Germany, 1994.

[6] M. R. Noraini és J. Geraghty, „Genetic Algorithm Performance with Different Selection Strategies in Solving TSP,” in *Proceedings of the World Congress on Engineering*, London, U.K., 2011.

[7] B. Czél and G. Gróf, "Inverse identification of temperature-dependent thermal conductivity via genetic algorithm with cost function-based rearrangement of genes," in *International Journal of Heat and Mass Transfer*, 2012.

[8] B. Czél és G. Gróf, „Genetic Algorithm-Based Method for Determination of Temperature-Dependent Thermophysical Properties,” in *International Journal of Thermophysics*, 2009.

[9] S. Mojrian, G. Pintér, J. H. Joloudari, I. Felde, L. Nádai, A. Mosavi és Á. Szabó-Gali, „Hybrid Machine Learning Model of Extreme Learning Machine Radial basis function for Breast Cancer Detection and Diagnosis; a Multilayer Fuzzy Expert System,” in *RIVF International Conference on Computing and Communication Technologies, pp. 1-7. , 7 p.*, 2020.

[10] J. Kennedy és R. Eberhart, „Particle swarm optimization,” *International Conference on Neural Networks,* %1. kötet4, pp. 1942-1948, 1995.

[11] L. Fung-Bao, "Particle Swarm Optimization-based algorithms for solving inverse heat conduction problems of estimating surface heat flux," in *International Journal of Heat and Mass Transfer*, 2012.

[12] P. Dousti, A. A. Ranjbar, M. Famouri és A. Ghaderi, „An inverse problem in estimation of interfacial heat transfer coefficient during two-dimensional solidification of Al

5%Wt-Si based on PSO," in *International Journal of Numerical Methods for Heat & Fluid Flow*, 2012.

[13] A. Janecek, Y. Tan és S. Zheng, „Enhanced Fireworks Algorithm," in *2013 IEEE Congress on Evolutionary Computation*, 2013.

[14] J. Li, S. Zheng és Y. Tan, „Adaptive Fireworks Algorithm," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014.

[15] Y. Z. Ying Tan, „Fireworks Algorithm for Optimization," in *Advances in Swarm Intelligence, First International Conference*, Beijing, China, 2010.

[16] Z. Fried, I. Felde és Á. Szabó-Gali, „A Fireworks Algoritmus kiterjesztése a komplex hőátadási együttható függvény rekonstrukciójához," in *A XXIX. Hőkezelő és anyagtudomány a gépgyártásban országos konferencia és szakkiállítás*, Balatonfüred, 2020.

[17] H. O. M.N. Özisik, „Inverse Heat Transfer: Fundamentals and Applications," *Taylor & Francis,* pp. 65-67, 1993.

[18] B. C. Verma S., „Multi-parameter estimation in combined conductione radiation from a plane parallel participating medium using genetic algorithms," *International Journal of Heat and Mass Transfer,* %1. kötet50, pp. 1706-1714, 2007.