

TDK DOLGOZAT

Barcsák Csaba

Intézmény: Eszterházy Károly Főiskola

Kar: Természettudományi Kar

Tanszék: Információtechnológiai Tanszék

Szerző: Barcsák Csaba

Témavezető: Biró Csaba tanársegéd

Konzulens: Prof. Dr. Jármái Károly, egyetemi tanár

Dolgozat címe:

Lennard-Jones atomklaszterek energiaminimumának meghatározása módosított
heurisztikus keresőalgoritmusokkal

Köszönetnyilvánítás

Köszönetemet fejezem ki témavezetőmnek, Biró Csaba tanár úrnak, aki figyelemmel kísérte, támogatta és építő javaslatokkal segítette munkámat. Köszönetemet fejezem ki konzulensemnek, Prof. Dr. Jármay Károly tanár úrnak, aki dolgozatom elolvasása után építő javaslatokat tett és évekkal ezelőtt bevezetett az optimalás kutatásának világába. Köszönetemet fejezem ki továbbá Dr. Hoffmann Miklós tanár úrnak és Sereg Enikőnek az építő javaslatokért.

Tartalomjegyzék

1. Bevezetés	1
2. Lennard-Jones atomklaszter feladat	2
2.1 A Lennard-Jones atomklaszter feladat megadása	3
3. PSO algoritmusok	4
3.1 Standard PSO algoritmus	4
3.2 PSO algoritmus bővítése	5
3.2.1 Bővítés gradiens becsléssel a részecskék pályája mentén	5
3.2.2 Bővítés lassított sebességű részecskékkel	6
3.3 Általános tesztproblémák	7
3.4 Numerikus eredmények	8
4. Minima Hopping	9
4.1 Minima Hopping algoritmus	9
4.2 A legjobb paraméterek kiválasztása	11
5. Konklúzió	17

1. fejezet

Bevezetés

Az optimalizációs problémák a tudomány sok területén megtalálhatóak. Ezek a problémák a célfüggvényekből és a feltételekből adódóan nagyon összetettek is lehetnek, egyes feladatokat analitikus módszerekkel nem, vagy csak nagyon nehezen lehet megoldani. Ezen feladatok közé tartozik az úgynevezett Lennard-Jones atomklaszter feladat is, ahol a rendszerben lévő atomok számának növekedésével a lokális szélsőértékek száma exponenciálisan nő. Az évek során sok különböző algoritmus született az ilyen problémák közelítő megoldására. A heurisztikus optimalizációs módszerek hatékonyak és viszonylag könnyedén implementálhatóak, emiatt nagy népszerűsége tettek szert az optimalizással foglalkozók körében, ezek közé az algoritmusok közé tartozik a PSO (Particle Swarm Optimization) és a Minima Hopping algoritmus is.

A dolgozatban olyan új algoritmusokat mutatunk be, melyek az eredeti PSO algoritmust több szempontból is továbbfejlesztik. Az egyik megközelítés véges differencia alapú gradiens becslést használ, nem igényel több függvénykiértékelést mint az eredeti algoritmus. A másik módszer az eddig megtalált legjobb eredményt pontosítja részecskék sebességének lassításával. Az új módszerek hatékonyak és könnyen implementálhatóak segítségükkel több magasabb változós számú irodalmi optimalizációs tesztprobléma is megoldásra került, a tesztek során láthatóvá vált, hogy az új technikák bizonyos tesztproblémák esetén nagyságrendekkel megnövelik az algoritmus hatékonyságát.

A módosított algoritmus hatékonysága a Lennard-Jones atomklaszter feladaton is teszteltük. A tesztek során megállapítottuk, hogy a bővítéseket használó PSO algoritmus hatékonyabb a bővítéseket nem használó változatnál, mivel pontosabban képes meghatározni a referenciaeredményt, az algoritmusok az atomok számának növekedésével valamelyest veszítenek hatékonyságukból.

A Minima Hopping algoritmus erős visszacsatolási mechanizmusa miatt jóval hatékonyabbnak bizonyult a Lennard-Jones atomklaszter feladat megoldására, mint a PSO algoritmusok. A kutatás során végrehajtott szimulációk során láthatóvá vált, hogy a Minima Hopping paraméterezése függ az atomok számától, mivel a globális szélsőértékhez közeli energiaszinteken változó a lokális szélsőértékek sűrűsége az energiaszintekre vonatkozóan. A 38 atomos rendszer esetében az irodalmitól eltérő paraméterezéssel az algoritmus hatékonyságát közel duplájára, a 100 atomos rendszer esetén pedig közel 20%-al sikerült megnövelnünk.

2. fejezet

Lennard-Jones atomklaszter feladat

A Lennard-Jones párpotenciálfüggvény [9] segítségével kémiai kötésben részt nem vevő atomok közötti kölcsönhatások modellezhetőek, emiatt használata elterjedtté vált a molekuláris mechanika területén belül. A függvényt az irodalom többféle formában írja fel, általános alakja a következő:

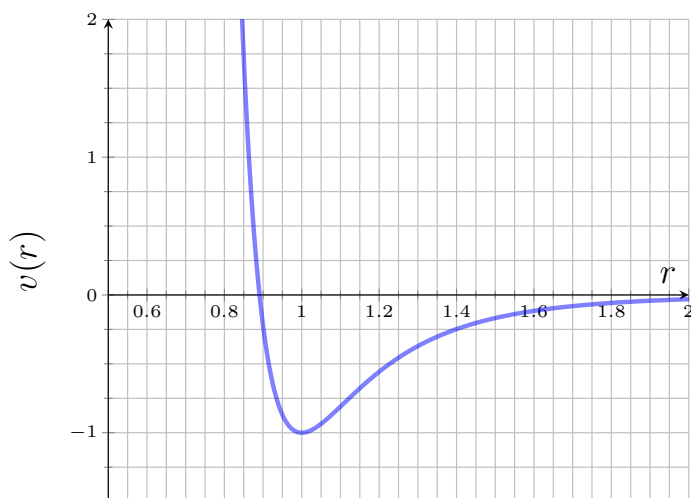
$$v(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right], \quad (2.1)$$

ahol ϵ a párpotenciálkút mélysége, r a két atom távolsága. A globális optimalás irodalma több publikáció esetén a következő formulát használja [11]:

$$v(r) = \frac{4}{r^{12}} - \frac{4}{r^6} \quad (2.2)$$

Ebben az esetben $\epsilon = \sigma = 1$. Egy másik megadási mód az ún. skálázott Lennard-Jones potenciál [10], ahol $\epsilon = 1$, $\sigma = 2^{-1/6}$. A dolgozatban ez a megadási mód került felhasználásra, a függvény grafikonja a 2.1. ábrán látható.

$$v(r) = \frac{1}{r^{12}} - \frac{2}{r^6} \quad (2.3)$$



2.1. ábra: Skálázott Lennard-Jones párpotenciálfüggvény

2.1. A Lennard-Jones atomklaszter feladat megadása

Jelölje $y^i = (y_1^i, y_2^i, y_3^i)^T$ az i . atom koordinátáit a háromdimenziós Euklideszi térben. Legyen $Y = ((y_1)^T, \dots, (y_N)^T)^T$, ahol N az atomok száma. Az i . és j . atom közötti Lennard-Jones potenciál:

$$v(r_{ij}) = \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^6}, \quad (2.4)$$

ahol

$$r_{ij} = \|y^i - y^j\| \quad (2.5)$$

Ekkor a Lennard-Jones atomklaszter feladat a következő formában írható fel:

$$\min_{Y \in \mathbb{R}^{3N}} V(Y) = \sum_{i < j} v(r_{ij}) \quad (2.6)$$

A 2.1. ábrán jól látható, hogy egy atompár esetén a potenciális energia minimuma -1, ekkor a két atom közötti távolság 1. A probléma formalizmusa egyszerű, és alacsony atomszámra az energiaminimum relatíve kevés számítással megtalálható, viszont a lokális szélsőértékek száma az atomok számával exponenciálisan nő, emiatt magasabb atomszám esetén a probléma nagyon nehezen megoldhatóvá válik. Léteznek olyan Lennard-Jones feladatok, melyeknek energiaminimumának meghatározása különösen nehéz, mivel az alacsony energiaszinten lévő elrendezések a konfigurációs térben távol vannak egymástól, ilyenek például a 38 és a 75 atomos rendszerek, melyeket a globális optimálással foglalkozó kutatók előszeretettel választanak tesztfeladatnak az algoritmusaik hatékonyságának ellenőrzésére.

Az irodalomban sok tanulmányt találunk a probléma heurisztikus algoritmusokkal történő megoldására, melyek a rendszer különféle empirikus vagy elméleti úton meghatározott paramétereinek felhasználásával (ilyen paraméter például az atomok közötti legkisebb távolság a rendszer energiaminimuma esetén [12]) igyekeznek az eddig megtalált referenciaeredményeknél [13] jobb eredményt adni, vagy gyorsabban meghatározni azokat. Ezen heurisztikus algoritmusok közé tartoznak módosított genetikai algoritmusok [14, 15, 16], a Basin Hopping [11], és a Minima Hopping [17] algoritmus is.

A Lennard-Jones atomklaszter feladat fontos probléma a fizikában, kémiában és a biológiában is, mivel gyors megoldásával olyan összetett energiaminimumon lévő rendszerek geometriáját ismerhetjük meg melyeknek kísérleti előállítása nagyon bonyolult [17].

3. fejezet

PSO algoritmusok

Kennedy és Eberhart 1995-ben mutatta be a PSO (Particle Swarm Optimization) algoritmust [1, 2], mellyel az eredeti céljuk az volt, hogy madarak csapaton belüli szociális viselkedését szimulálják és vizualizálják. A kutatásaik során felfedezték, hogy ez a módszer optimálási feladatok megoldására is hatékonyan alkalmazható. A PSO első változata csak folytonos nemlineáris optimálási feladatokat volt képes megoldani. Az évek során az algoritmusnak nagyon sok változata jelent meg, melyek optimálási problémák széles skáláját képesek megoldani. Ezen algoritmusok az egyszerűségük és hatékonyságuk miatt váltak széles körben elterjedtté a mérnöki gyakorlatban [3, 4, 5].

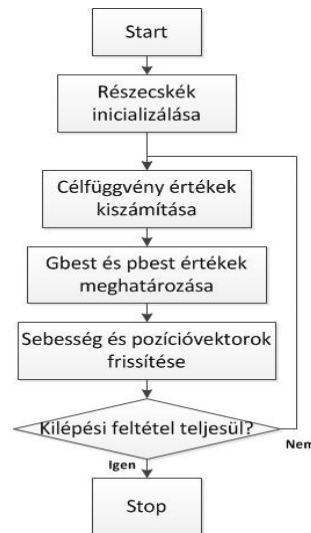
3.1. Standard PSO algoritmus

A következőkben a standard algoritmus kerül ismertetésre. Az algoritmus első lépésben ún. részecskéket generál. Minden részecske rendelkezik egy x pozíció, és egy v sebességvektorral. Ezen vektorok elemszáma megegyezik a célfüggvény változóinak számával. A helyvektorok generálása a célfüggvény előre definiált tartományán egyenletes eloszlás szerint történik. A részecskék a megadott tartományon mozognak, és keresik az optimális megoldást. Minden részecske tárolja a mozgása során talált legjobb megoldást és annak pozícióját, ezeket lokális legjobb néven említi az irodalom. Külön tárolásra kerül a lokális legjobbak közül a legjobb. Ezt nevezzük globális legjobbnak. A részecskék minden iterációs lépésben újabb mintát vesznek a célfüggvényből, valamint változtatják a pozíciójukat és sebességüket a következő egyenletek szerint.

$$v_i^{k+1} = v_i^k + c_1 r_1 (pbest_i - x_i^k) + c_2 r_2 (gbest_i - x_i^k) \quad (3.1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} t \quad (3.2)$$

Ahol v_i a sebességvektor i -edik eleme, x_i a pozícióvektor i -edik eleme, c_1 és c_2 pozitív konstansok, r_1 és r_2 két egyenletes eloszlás szerint generált véletlen szám a $[0,1]$ intervallumon, $pbest_i$ az adott részecske lokális legjobb pozíciójának i -edik eleme $gbest_i$ pedig a globális legjobb pozíció i -edik eleme, a k index az adott iterációt jelöli, t az egységnyi időintervallum. Az algoritmus lépéseit a 3.1. ábra mutatja:



3.1. ábra: Standard PSO algoritmus folyamatábrája

A 3.1. ábrán látható folyamatban a részecskék inicializálása a helyvektoraik inicializálásaként értendő. Ezen vektorok egyenletes eloszlás szerinti véletlen értékeket kapnak, ügyelve arra, hogy a helyvektorok ne legyenek az előre definiált tartományon kívül. A PSO algoritmus amiatt lett népszerű, mert működése könnyen megérthető, egyszerűen implementálható, könnyen integrálható más optimáló eljárásokba, kevésbé érzékeny célfüggvényre, beállításához kevesebb paraméter szükséges, mint más heurisztikus eljárásoknál. A hátránya, hogy nincs mögötte mély matematikai háttér.

3.2. PSO algoritmus bővítése

Az előbbieken említésre került, hogy a PSO algoritmusnak sokféle változata jelent meg, hatékonyságának javítására többféle technikát találhatunk az irodalomban. Az egyik ismert megoldás, hogy egyidejűleg több részecskecsoporttal dolgozunk az egy helyett, ekkor tároljuk részecskénként a lokális legjobb eredményt, az egyes csoportokhoz tartozó legjobb eredményt, és a részecskecsoportok legjobbjai közül a legjobbat. Ekkor nemcsak az egyes részecskék között, hanem a részecskecsoportok között is értelmezett a kommunikáció, tehát az egyes részecskéknek a sebesség és pozíció változtatásánál, a lokális legjobb, a csoport legjobb, és a csoportok összességének legjobb eredményeit is figyelembe vesszük. Egy másik megoldás az ún. crazy bird módszer. Ez a megoldás véletlenszerűen kiválaszt részecskéket, és ezen részecskék sebességét nem a 3.1. formula alapján változtatjuk, hanem véletlenszerű irányba, ezáltal kiszakítunk részecskéket a csoportból melyek nem a gbest irányába tartanak, remélve ezzel azt, hogy más irányba jobb eredményt találunk mint a jelenlegi gbest.

3.2.1. Bővítés gradiens becsléssel a részecskék pályája mentén

Az előbbieken említett eljárások hatékonysága nagyban függ a véletlentől. A standard algoritmus a célfüggvény kiszámított értékein kívül más információkkal nem rendelkezik a függvényről, pedig több esetben hasznos lenne a függvény egyes lokális tulajdonságainak ismerete, mivel ezen információk nem kötődnek a véletlenhez és általuk

hatékonyabbá tehetnénk az eljárást. Az egyik ilyen lokális tulajdonság a gradiens, melyet, mivel csak diszkrét pontokban rendelkezünk mintákkal, becsülnünk kell. A véges differencia alapú megoldások gyors és hatékony megoldást nyújtanak a gradiens becslésére diszkrét adatok esetén.

Egy adott részecske mozgása során adott pillanatig érintett pontokhoz tartozó függvény értékeket felhasználhatjuk az adott pontokban vett részecske pályájának menti gradiensek becslésére. Az általunk implementált algoritmusban a backward difference [18] módszer került beépítésre az egyszerű implementálhatósága miatt. A gradienseket az elkészült algoritmus a részecskék sebességének beállítására használja, ezáltal a részecskék a függvény értelmezési tartományának egyes intervallumaiban gyorsabban, míg más intervallumokban lassabban mozognak. Minden egyes részecske tárolja, hogy a már érintett pontokban hány egymás utáni esetben talált pozitív előjelű gradienst. A pozitív előjelű gradiensek számának a függvényében a részecskét lineárisan gyorsítjuk, ha negatív előjelű gradienst találunk, vagy nem értelmezett a gradiens, akkor a sebesség visszaáll az alapértékre. Ha egy adott részecske esetén nagy az egymás utáni mintavett pontokban egymást követő pozitív előjelű gradiensek száma, abból az következik, hogy a részecske ezen idő alatt nem haladt át lokális szélsőértéken, emiatt vagy a globális szélsőérték felé tart vagy egy olyan lokális szélsőértékhez, amely a részecske környezetében egy nagyobb intervallumon vett szélsőérték, emiatt gyorsíthatunk a sebességen, hogy a részecske kevesebb iterációs lépés alatt elérjen a szélsőértékig. A célfüggvény változójának növekedése esetén a bővítés nem igényel több függvénykiértékelést, mint ahogyan az irodalmi megoldások esetén látható [19, 20], mivel a gradienst csak a részecske pályájának iránya mentén becsüli, ami minden esetben egy görbe.

3.2.2. Bővítés lassított sebességű részecskékkel

A részecskék pályájának mentén vett gradiens becslés segítségével megnövelhetjük az algoritmus konvergenciájának a sebességét [22], viszont egy hatékony módszerhez nemcsak az algoritmus feltérképező tulajdonságát, hanem az eddig megtalált legjobb eredmény pontosítását is erősítenünk kell. Az előbbieken ismertetett módszert felhasználva az egyik lehetséges megoldás az, ha a részecskék pályája mentén vett gradiens előjelet vált, akkor előre definiált valószínűséggel csökkentésre kerül az adott részecske sebessége előre definiált iteráció számon át. Az új ötletet a multi swarm megközelítés ihlette [21]. A módszer amiatt előnyös, mert így jobban feltérképezésre kerül az adott lokális szélsőérték. Az előre definiált iterációs szám, ami után a részecske elveszíti a csökkentett sebesség tulajdonságát a módszer feltérképezési tulajdonságát erősíti, mivel egy lokális optimum feltérképezése után a részecske tovább folytathatja útját egy esetlegesen megtalált jobb megoldás felé. Abban az esetben, ha a gradiens előjel váltása esetén a populáció méretéhez képest sok részecske kerül a lassított állapotba, akkor az adott lokális szélsőérték jobban feltérképezésre kerül, viszont az algoritmus feltérképez tulajdonságából veszítünk, emiatt a megfelelő paraméterezés problémánként változhat. Azt tapasztaltuk, hogy a módszer akkor találja meg kevesebb iteráció alatt jobb közelítéssel a referenciaeredményt, ha a lassított sebességű részecskék száma nem haladja meg a folyamatban részt vevő részecskék számának a felét, az egy iteráció alatt lassításra kerülő részecskék számát pedig érdemes alacsonyan tartani, mivel ha egyszerre sok részecske lassul, akkor az újonnan megtalált legjobb eredménynek csak a közvetlen környezete kerül feltérképezésre.

3.3. Általános tesztproblémák

Ebben az alfejezetben az algoritmusok tesztelésre szolgáló tesztfüggvények kerülnek ismertetésre. Sokféle tesztproblémát és eredményt találhatunk az irodalomban [6, 7, 8], viszont a szerzők ritkán írnak arról, hogy az adott tesztfeladatot miért választották be a saját keretrendszerükbe. A tesztfeladatok nehézsége függ a lokális szélsőértékek számától és eloszlásától, emellett a függvény változóinak számától is. A keretrendszer kialakítása során figyelembe vettem a heurisztikus kereső algoritmusok leggyakoribb hibáit. A következő táblázatban öt irodalomból kiválasztott tesztfeladat és két saját tesztprobléma kerül ismertetésre, melyeknek a globális szélsőértékei ismertek.

Név	Definíció	Optimum (maximum)
<i>De Jong</i>	$f(\mathbf{x}) = -\sum_{i=1}^n x_i^2$	$-100 \leq x_i \leq 100$ $\mathbf{x}^* = (0, 0, 0, \dots, 0)$ $f(\mathbf{x}^*) = 0$
<i>Modified Generalised Easom</i>	$f(\mathbf{x}) = \left[\text{abs} \left(\prod_{i=1}^n [\cos(x_i)] e^{-\sum_{i=1}^n \frac{1}{5} x_i^2} \right) \right]^{0.1}$	$-10 \leq x_i \leq 10$ $\mathbf{x}^* = (0, 0, 0, \dots, 0)$ $f(\mathbf{x}^*) = 1$
<i>Rosenbrock</i>	$f(\mathbf{x}) = -\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-2 \leq x_i \leq 2$ $\mathbf{x}^* = (1, 1, 1, \dots, 1)$ $f(\mathbf{x}^*) = 0$
<i>Generalised Drop Wave</i>	$f(\mathbf{x}) = \frac{1 + \cos(12\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2})}{\frac{1}{2}(x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2) + 2}$	$-2 \leq x_i \leq 2$ $\mathbf{x}^* = (0, 0, 0, \dots, 0)$ $f(\mathbf{x}^*) = 1$
<i>Rastrigin</i>	$f(\mathbf{x}) = -(10n + \sum_{i=1}^n (x_i^2 - 10\cos(2\pi x_i)))$	$-5 \leq x_i \leq 5$ $\mathbf{x}^* = (0, 0, 0, \dots, 0)$ $f(\mathbf{x}^*) = 0$
<i>Two Towers</i>	$f(\mathbf{x}) = \prod_{i=1}^n [\cos(3(x_i - 5))] e^{-(\sum_{i=1}^n 3(x_i - 5))^{0.2}} + 1.2 \prod_{i=1}^n [\cos(3(x_i - 5))] e^{-(\sum_{i=1}^n 3(x_i - 5))^{0.5}}$	$-10 \leq x_i \leq 10$ $\mathbf{x}^* = (-5, -5, -5, \dots, -5)$ $f(\mathbf{x}^*) \approx 1.2$
<i>Four Drop Waves</i>	$f(\mathbf{x}) = \sin\left(\frac{1}{5} \prod_{i=1}^n x_i\right)^2 \frac{1}{10^n} \text{abs}\left(\prod_{i=1}^n x_i\right) + 1.1 \prod_{i=1}^n [\cos(x_i)] e^{-\sum_{i=1}^n x_i^2}$	$-10 \leq x_i \leq 10$ $\mathbf{x}^* = (0, 0, 0, \dots, 0)$ $f(\mathbf{x}^*) = 1.1$

3.1. táblázat: Tesztproblémák heurisztikus keresőalgoritmusok vizsgálatához

A De Jong, Rosenbrock és Modified Easom függvények kevés lokális szélsőértékkel rendelkeznek, ezek segítségével alacsony változó szám mellett megbizonyosodhatunk róla, hogy az algoritmusunk megfelelően működik és nem vétettünk implementációs hibát. A függvények változószámát növelve jól láthatóvá válik, hogy az algoritmus mennyire tudott közel kerülni az optimumhoz, mivel a lokális szélsőértékek kevésbé befolyásolják az eredményt. A Generalized Drop Wave és a Rastrigin nehezebb problémák, melyek sok lokális szélsőértékkel rendelkeznek. A Two Towers és a Four Drop Waves saját készítésű tesztfüggvények, melyek az előzőekben említetteknél is több hívást jelentenek az algoritmus számára.

3.4. Numerikus eredmények

A PSO változatok tesztelésre kerültek az előző alfejezetben leírt tesztproblémákra és a Lennard-Jones feladatra is, az eredményeket a következő táblázatok mutatják:

		Változók száma								
		2	3	4	5	6	7	8	9	10
De Jong	PSO	0.00013	0.00420	0.04771	0.26054	1.05916	3.13830	6.77713	14.35504	25.54344
	GPSO	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00008	0.00043	0.00179
Modified	PSO	0.00000	0.00000	0.00004	0.00028	0.00094	0.00284	0.00752	0.02638	0.04383
	GPSO	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00814
Rosenbrock	PSO	0.00000	0.00030	0.06773	1.27169	3.36706	4.75454	6.42374	7.57780	9.17560
	GPSO	0.00000	0.00000	0.01210	0.63283	2.42346	4.19583	5.91742	7.10272	8.44196
Generalised	PSO	0.00000	0.00072	0.01624	0.05674	0.06375	0.06375	0.06375	0.06375	0.06375
	GPSO	0.00000	0.00000	0.02486	0.05355	0.06375	0.06375	0.06375	0.06375	0.06375
Drop Wave	PSO	0.00414	0.12960	0.56275	1.39025	2.22276	3.45297	4.92020	6.08466	8.07402
	GPSO	0.00000	0.06626	0.40299	0.98149	2.06641	2.89886	3.78870	4.99043	5.73451
Rastrigin	PSO	0.00034	0.17921	0.34870	0.61217	0.95946	1.10017	1.14179	1.16189	1.17427
	GPSO	0.00026	0.13667	0.22520	0.36127	0.88934	1.07802	1.13582	1.16380	1.17480
Four	PSO	0.00000	0.01211	0.12198	0.19751	0.29189	0.44158	0.58276	0.69408	0.80301
	GPSO	0.00000	0.01515	0.11217	0.18462	0.27792	0.41077	0.53980	0.65740	0.81365

3.2. táblázat: PSO algoritmusok eredményei általános tesztproblémákon. A tesztfüggvényeken 1000 iterációig futottak 100 részecskével a PSO változatok, minden változószámmra minden függvény esetén 100-szor. A táblázatban az elméleti megoldás és az algoritmusok által megtalált megoldás különbségének abszolút értéke látható átlagolva a 100 futásra. A PSO a standard algoritmust, míg a GPSO a gradiens becsléssel és részecske lassítással ellátott változatot jelöli. A csökkentett sebesség 50 iteráción át tart. Iterációnként átlagosan egy részecske léphet be a csökkentett sebességűek közé, a csökkentett sebességű részecskék száma a teljes részecskeszám legfeljebb fele lehetett a szimulációk során.

A táblázat alapján látható, hogy a bővített algoritmus nagyságrendekkel jobban teljesített a bővítéseket nem tartalmazónál a kevés lokális szélsőértékkel rendelkező függvények esetén (De Jong és Modified Generalised Easom) magasabb változószám esetén is. A többi esetben láthatóan jobb eredmények 5 változó alatt születtek, 5 változó felett kisebb a javulás mértéke, emiatt a bővítések alacsony változószám esetén javasolhatóak.

		Atomok száma			
		2	3	4	5
Lennard-Jones feladat	PSO	0.00000	0.00000	0.23797	1.03045
	GPSO	0.00000	0.00000	0.18266	0.75490

3.3. táblázat: Az algoritmusok az előző táblázatban ismertetett paraméterekkel futottak A Lennard-Jones feladatra. A táblázatban az elméleti megoldás és az algoritmusok által megtalált megoldás különbségének abszolút értéke látható átlagolva a 100 futásra. A bővített változat jobban teljesített, de a 4 és 5 atomos rendszerek esetén már nem minden esetben talált megoldást sem a standard, sem a bővített változat, mivel a dinamikai rendszer veszít a hatékonyságából magas változószám esetén.

4. fejezet

Minima Hopping

A Minima Hopping [17, 16, 23, 24, 25] speciálisan energiaminimálási problémákra kidolgozott algoritmus, amely a Bell-Evans-Polányi elven alapszik. A módszer ugrásokat hajt végre a szélsőértékek között, és a megtalált új minimumokról thresholding segítségével dönti el, hogy elfogadja-e azokat. A threshold értéke a szimuláció során úgy változik, hogy a megtalált szélsőértékek körülbelül fele kerüljön elfogadásra.

4.1. Minima Hopping algoritmus

Az algoritmust logikailag két fő részre lehet bontani. Az első logikai részben ugrásokat hajt végre, hogy eltávolodjon az eddig megtalált szélsőértékektől és újabbakat találjon meg, ezt lokális optimálás követi, mely pontosítja az eredményt. Az ugrásokat az algoritmus molekuladinamikai szimuláció segítségével oldja meg, a lokális optimálásra pedig a steepest descent és konjugált gradiens módszereket használják az irodalomban. Az atomok sebességvektorai Boltzmann eloszlás szerint kerülnek generálásra. A szimuláció akkor ér véget, amikor átugrott $mdmin$ darab maximumot és minimumot. Ezután indul be a lokális optimálós. Az algoritmus addig hajt végre molekuladinamikai szimulációkat különféle Boltzmann eloszlás szerint generált sebességvektorokkal, majd utána lokális optimálást, amíg egy eddig fel nem térképezett minimumot talál. Az első rész kimenetele többféle lehet. Az első eset az, amikor a jelenlegi minimumot találjuk meg ismételten. A második eset, ha egy már régebben megtalált minimumot találunk meg, a harmadik eset az, ha eddig még meg nem talált minimumra akadunk. Az első két esetben az algoritmus megnöveli a szimulációnál használt kinetikus energiát változtatva, ezzel az atomok kezdeti sebességvektorainak hosszát, hogy a szimuláció eddig még feltérképezetlen régiók felé vigye a rendszert. Új minimum megtalálása esetén a potenciális energia csökkentésre kerül. A második logikai részben történik az újonnan megtalált minimum elfogadása vagy eldobása thresholding segítségével, ahol a jelenlegi és az újonnan megtalált minimum potenciális energiájának különbségét veszi alapul az algoritmus. Ha a minimum eldobásra kerül, akkor a rendszer növeli a threshold értéket, ezáltal nagyobb valószínűséggel fogad majd el olyan új minimumot ahol a jelenlegi és az új minimum potenciális energiájának nagyobb a különbsége. Ha elfogadja az új minimumot, akkor csökken a threshold érték. Ez a logika azt a célt szolgálja, hogy az algoritmus kisebb ugrásokat hajtsa végre, felhasználva ezzel a Bell-Evans-Polányi elvet, ami szerint kisebb ugrásokkal nagyobb valószínűséggel kerül a rendszer alacsonyabb energiaszintre, mint nagyobb ugrásokat használva.

```

Input inicializálása, melynek minimuma  $M_{Current}$ ;
MD szimuláció indítása  $M_{Current}$  -ból;
Amint a szimuláció eléri az  $mdmin$ -edik minimumot
lokális optimáló indítása amely megtalálja az  $M$  minimumot;
if  $M == M_{Current}$  then
    |  $E_{kinetic} = E_{kinetic} * \beta_S$ ;
    | goto MD szimuláció indítása;
end
if  $M \in History$  then
    |  $E_{kinetic} = E_{kinetic} * \beta_O$ ;
    | goto MD szimuláció indítása;
end
 $E_{kinetic} = E_{kinetic} * \beta_N$ ; // Új minimumot találtunk;
if  $(M - M_{current}) < Ediff$  then
    |  $M_{current} = M$ ; // Új minimum elfogadása;
    |  $M_{current}$  hozzáadása a History-hoz;
    |  $Ediff = Ediff * \alpha_A$ ;
else
    |  $Ediff = Ediff * \alpha_R$ ;
end
goto MD szimuláció indítása;

```

4.1. ábra: Minima Hopping algoritmus

Az algoritmus hatékonysága nagyban függ a szimulációtól és a fenti pszeudokódban látható 5 fő paramétertől. A β_S , β_O és β_N paraméterek a kinetikus energia változásának sebességét befolyásolják. $\beta_S > 1$ és $\beta_O > 1$ mivel ezek segítségével növeljük a kinetikus energiát és $\beta_N < 1$, mivel új minimum megtalálása esetén csökkentjük. Az α_A és α_R paraméterek az $Ediff$ értékét befolyásolják az új minimum elfogadása vagy eldobása esetén, $\alpha_A < 1$ és $\alpha_R > 1$. Az irodalom kétféle paraméterbeállítást javasol [17, 25]. Az egyik $\beta_S = \beta_O = 1/\beta_N = 1.05$ és $\alpha_R = 1/\alpha_A = 1.02$ illetve a $\beta_S = \beta_O = 1/\beta_N = \alpha_R = 1/\alpha_A = 1.05$. Az irodalomban olvashatunk továbbá egy fejlesztett visszacsatolási mechanizmusról [16] ahol $\beta_O = \beta_O^0(1 + c \cdot \log(N))$, $\beta_O^0 = 1.05$, $c = 0.1$ és N az adott minimumhoz történő eddigi visszatérések száma. Ez a módszer nagy atomszám esetén gyorsítja a keresést, mivel az algoritmus ezáltal gyorsabban képes kiugrani a már feltérképezett régióból. Egy másik bővítési lehetőség ún. softening [16], amely a molekuladinamikai szimuláció kezdetekor a sebességvektorok irányát alacsony görbületű régiók felé irányítja, emiatt nagyobb valószínűséggel következik be kisebb ugrás, jobban kihasználva ezzel a Bell-Evans-Polányi elvet.

4.2. A legjobb paraméterek kiválasztása

Az irodalomban a már előbb említett kétféle paraméterezés jelenik meg, a szerzők azt írják, hogy ezekkel a paraméterekkel az algoritmus mindig megtalálta a referenciaeredményt, választhatunk más paramétereket is, ebben az esetben lehet gyorsabb lesz az algoritmus, viszont az is előfordulhat, hogy nem találjuk meg a referenciaeredményt. Ez a kijelentés több kérdést is felvet. Milyen paraméterek esetén lesz gyorsabb az algoritmus anélkül, hogy a módszer a hatékonyságából veszítene? Melyik a legjobb paraméter választás és hogyan függnek a paraméterek az aktuális rendszertől? Dolgozatomban ezekre a kérdésekre is kerestem a választ. Ahhoz, hogy közelebb kerüljünk a válaszokhoz, érdemes szimulációkat végrehajtani többféle paraméterezéssel és megvizsgálni a futásidőt és azt, hogy az algoritmus megtalálja-e a referenciaeredményt. A vizsgált paraméterbeállításokat a következő táblázat mutatja:

Beállítások elnevezése	Paraméterek				
	β_S	β_O	β_N	α_A	α_R
Set 0	1.045	1.045	1/1.045	1/1.02	1.02
Set 1 [17]	1.05	1.05	1/1.05	1/1.02	1.02
Set 2 [25]	1.05	1.05	1/1.05	1/1.05	1.05
Set 3	1.1	1.1	1/1.1	1/1.1	1.1
Set 4	1.15	1.15	1/1.15	1/1.15	1.15
Set 5	1.2	1.2	1/1.2	1/1.2	1.2

4.1. táblázat: Paraméter beállítások Minima Hopping algoritmus vizsgálatához

A szimulációhoz szimmetrikus paraméterek kerültek kiválasztásra, hogy a paraméterek megnövelt értékeinek a hatását tudjuk vizsgálni. A növelés az 1-nél nagyobb értéket felvevő paraméterekre értendő. Ha egy 1-nél nagyobb paramétert megnövelek, akkor az 1-nél kisebb "párját" csökkentem, így ugyanazzal a számmal szorzom például a kinetikus energiát egy régi eredmény megtalálásakor, mint amivel elosztom ha az algoritmus új eredményt talált. Ha ez nem kerülne bevezetésre, akkor nagyobb valószínűséggel alakulna ki olyan helyzet, hogy a rendszer kinetikus energiája robbanásszerűen megnő mielőtt a módszer megtalálná az optimumot. Természetesen nem szimmetrikus paraméterek esetén is működhet az algoritmus, viszont ezekkel további nemlineáris hatásokat is bevinnénk a rendszerbe az előbb említett probléma mellett, melyek megnehezítenék az eredmények értékelését, ez egy további kutatás célja lehet. A szimulációk során a $mdmin = 4$ beállítással dolgoztunk, felhasználásra került a fejlesztett visszacsatolás és a softening. A referenciaeredmény megtalálását 5 tizedesjegy pontossággal vártuk el az algoritmustól.

Beállítások	Atomok száma									
	20	30	38	40	50	60	70	80	90	100
Set 0	0.227	20.806	50.248	3.618	10.940	18.874	92.239	119.387	489.906	650.733
Set 1	0.132	19.071	49.284	3.760	13.328	18.509	78.767	116.757	474.277	796.535
Set 2	0.133	13.518	53.049	3.167	13.449	21.223	84.088	106.665	590.776	926.664
Set 3	0.127	10.178	31.305	3.396	10.826	17.667	84.827	102.226	534.929	823.218
Set 4	0.163	6.900	22.154	3.239	9.004	18.437	88.684	109.336	600.090	1081.387
Set 5	0.135	7.616	28.564	3.592	11.122	20.823	84.195	122.545	581.043	1102.710

4.2. táblázat: Átlagos futásidő (CPU time másodpercekben) különböző atomszámokra különféle paraméterbeállításokkal. Az algoritmus minden atomszámra és minden egyes paraméterbeállításra százszor lett futtatva Intel Core i7 3630QM CPU-n párhuzamosítás nélkül. A Set 0 esetén 5, Set 1 esetén 2, míg a Set 2 esetén 3 esetben nem találta meg a módszer a referenciaeredményt az összes atomszámra vonatkozóan, az összes többi esetben megtalálta a referenciaeredményt. Ennek az az oka, hogy alacsony atomszám esetén az algoritmus 10000 megtalált szélsőértékig futtatott, ha addig nem talált eredményt, akkor nem volt sikeres a keresés az adott futás esetén. A különféle színek egy bizonyos atomszám esetén az eredmények sorrendjét jelölik (a sötétzöld a legkisebb futásidő, a sötétpiros a legnagyobb).

A táblázat színezése alapján látható, hogy az irodalminál magasabb paraméterekkel dolgozó paraméterbeállítások (Set 3,4,5) az esetek nagy részében jobban teljesítettek 60 atom alatti rendszerek esetén, magasabb atomszám esetén pedig a Set 0 és az irodalmi megoldások bizonyultak jobbnak több esetben. Az irodalmitól eltérő paraméterekkel jelentősen nagy javulást a 30 és 38 atomos rendszerek esetében látni illetve Set 0 esetén 100 atomra, 38 atomra paraméterbeállításokként 1000 futás eredményeit vizsgáltuk meg beállításokként a következő paraméterekkel:

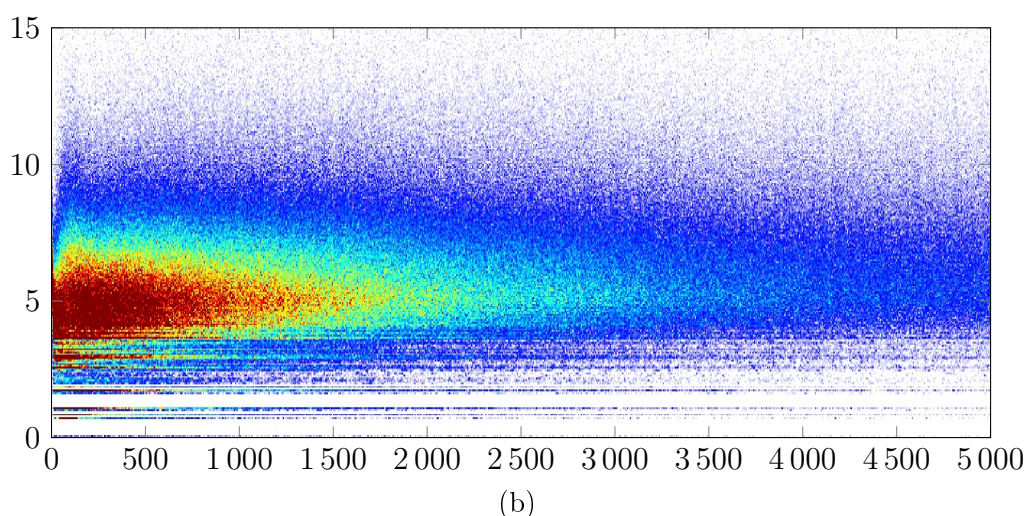
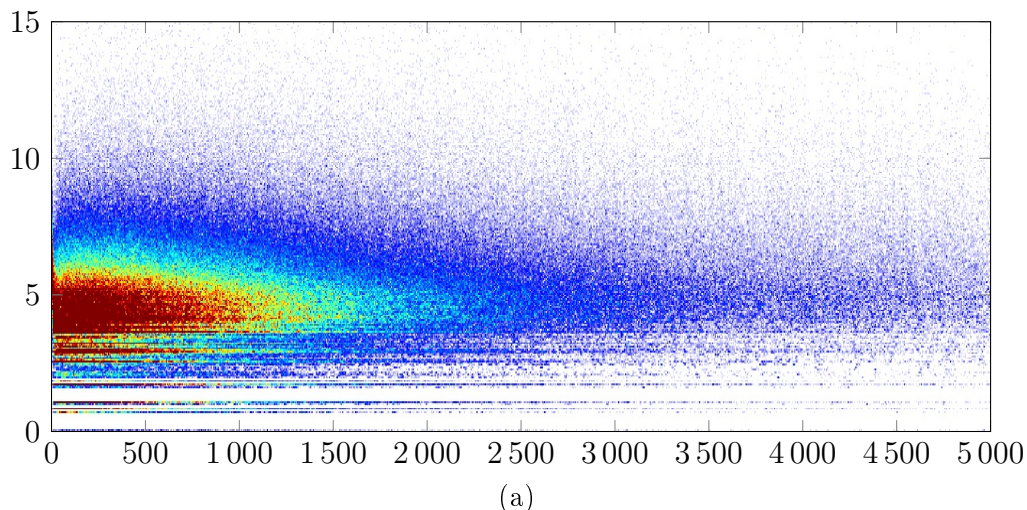
Beállítások elnevezése	Paraméterek				
	β_S	β_O	β_N	α_A	α_R
LJ38 Set 1 [17]	1.05	1.05	1/1.05	1/1.02	1.02
LJ38 Set 2 [25]	1.05	1.05	1/1.05	1/1.05	1.05
LJ38 Set 3	1.1	1.1	1/1.1	1/1.1	1.1
LJ38 Set 4	1.15	1.15	1/1.15	1/1.15	1.15
LJ38 Set 5	1.155	1.155	1/1.155	1/1.155	1.155
LJ38 Set 6	1.16	1.16	1/1.16	1/1.16	1.16

4.3. táblázat: Paraméter beállítások Minima Hopping algoritmus vizsgálatához 38 atomos rendszer esetén

LJ 38		
Beállítás	Átlagos futásidő	Találati arány
LJ38 Set 1	49.575	97.0%
LJ38 Set 2	46.039	97.1%
LJ38 Set 3	30.071	99.5%
LJ38 Set 4	26.338	100%
LJ38 Set 5	26.877	100%
LJ38 Set 6	27.955	100%

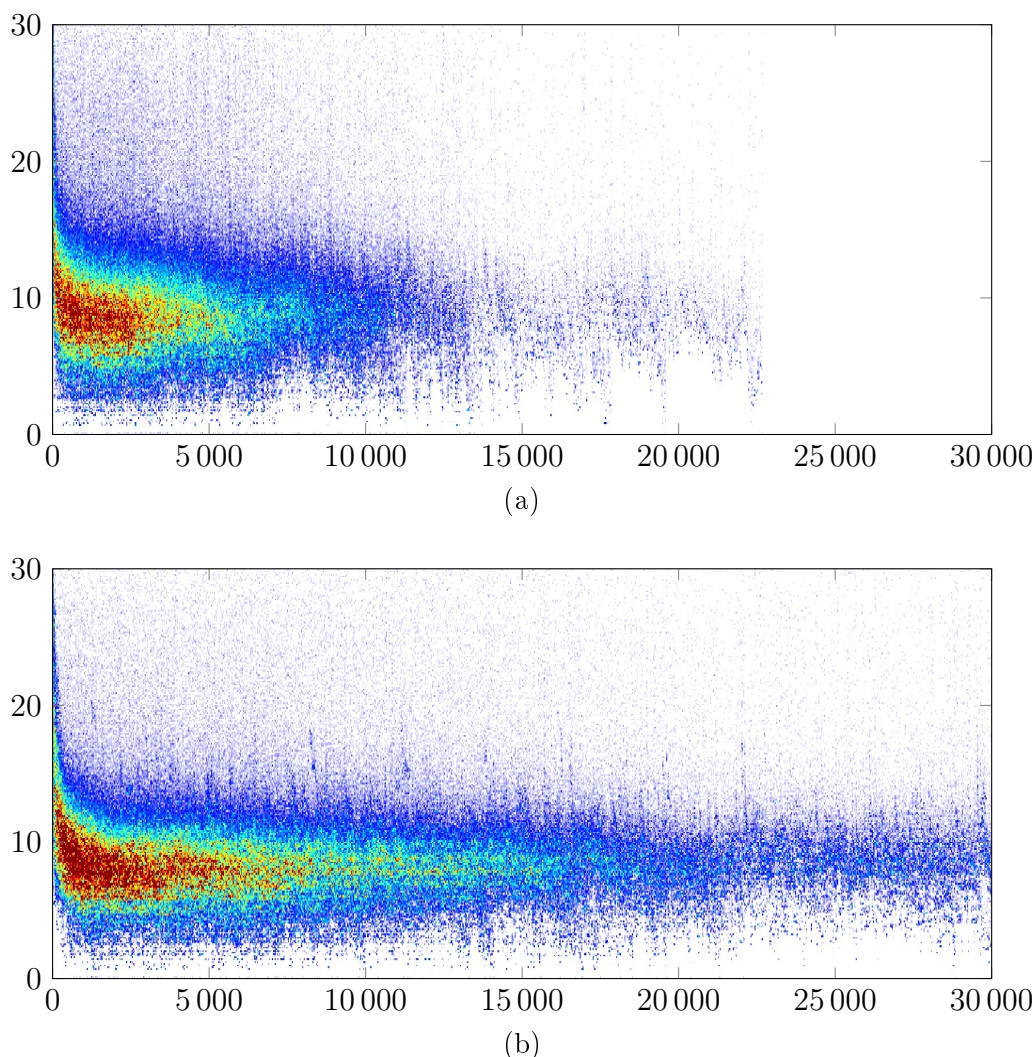
4.4. táblázat: Átlagos futásidő (CPU time másodpercekben) LJ38 esetén különféle paraméter-beállításokkal Intel Core i7 3630QM CPU-n párhuzamosítás nélkül. Látható, hogy az eredmények paraméterbeállításokként 1000 futtatás esetén is jobbak az új paraméterezéssel, és az is, hogy a legjobb futásidőt a LJ38 Set 4 beállítás közelében találjuk 38 atomos rendszer esetén. A találati arány kisebb az irodalmi paraméterezés esetén, ennek az az oka, hogy a szimuláció során az algoritmus 10000 megtalált szélsőértékig futtatott, ha addig nem talált eredményt, akkor nem volt sikeres a keresés az adott futás esetén.

Az eredmények alapján látható, hogy kisebb atomszámú rendszerek esetén az irodalmi megoldások rosszabbul teljesítenek, és magasabb atomszám esetén jobban. Mi lehet ennek az oka? A 4.2-es táblázat alapján kirajzolódik egy trend arra vonatkozóan, hogy melyik paraméterezés a jobb az atomszám függvényében, viszont összevetve az adatokat a 38 atomos rendszert beállításokként 1000 futással vizsgáló 4.4-es táblázattal, észrevehető, hogy a 4.2-es tartalmaz némi zajt, a 100 futás helyett érdemes lenne minden esetben 1000 futást vizsgálni több paraméterbeállítással, magasabb atomszámú rendszerekre is, viszont ahhoz egy PC-nél nagyobb hardware erőforrásra van szükség. Látható továbbá az is, hogy LJ38 esetén a LJ38 Set 4 beállítás teljesített a legjobban és a LJ38 Set 1 a legrosszabbul az 1000 futást vizsgáló szimulációban, és LJ 100 esetén pedig a Set 0 a legjobban és Set 5 a legrosszabban teljesítő. Arra keressük a választ, hogy a legjobban teljesítő paraméterezés esetén a futásokban mi a közös, illetve a legrosszabbul teljesítő paraméterezés esetén a futásokban mi a közös. Ha erre a kérdésre tudjuk a választ, akkor tudunk előre lépni. Vizsgáljuk meg a következő density plotokat:



4.2. ábra: A képeken density plotokat látunk. A vízszintes tengely mentén a megtalált minimum sorszáma (0-tól kezdve a számozást), a függőleges tengely mentén pedig a potenciális energiaszint (a referenciaeredmény 0-ba lett eltolva). A density plot tartalmazza az adott paraméterbeállítás során megtalált összes minimumot az 1000 futásra az a) ábrán a LJ38 Set 4 és a b) ábrán a LJ38 Set 1 beállítások esetén. A fehér szín azt jelöli, hogy az adott régióban az algoritmus nem végzett feltérképezést, a legsötétebb piros szín pedig azt, hogy az adott pixelen (minden pixel az vízszintes tengely mentén 0.036, a függőleges tengely mentén pedig 3.3 egységen vizsgálja a gyakoriságot) 50 a feltérképezett minimumok gyakorisága. A kettő közötti átmenetet különböző kék és sárga színek jelölik.

Az ábrák jól szemléltetik a folyamatot, többet mondanak annál, mint ha egyetlen futás során megtalált lokális minimumokat jelenítenénk meg (mivel sok esetben eléggé nagyok az ugrások, nem lenne informatív az ábra) és a pusztán futás idő számítani átlagnál is többet mond. Nézzük meg a vízszintes tengelyen az 5000-edik megtalált minimum környezetét. Az a) ábrán kevesebb a kék pont, mint a b) ábrán, mivel az a) ábrán látható LJ38 Set 4 beállítással a módszer ekkorra sok esetben már megtalálta a megoldást. Látható továbbá, hogy mindkét esetben a függőleges tengely mentén 4 egység közelében találhatóak a leginkább látogatott energiaszintek, emellett mindkét esetben a függőleges tengely mentén 2 egység alatt vannak olyan régiók, amit a módszer nem térképezett fel. Ez amiatt lehet, mert az alacsonyabb energiaszintre jutáshoz nagyobb ugrásokra volt szükség. Vizsgáljuk meg a LJ100-as rendszert.



4.3. ábra: A képeken density plotokat látunk. A vízszintes tengely mentén a megtalált minimum sorszáma (0-tól kezdve a számozást), a függőleges tengely mentén pedig a potenciális energiaszint (a referenciaeredmény 0-ba lett eltolva). A density plot tartalmazza az adott paraméterbeállítás során megtalált összes minimumot az 100 futásra az a) ábrán a Set 0 és a b) ábrán a Set 5 beállítások esetén 100 atomos rendszerre. A fehér szín azt jelöli, hogy az adott régióban az algoritmus nem végzett feltérképezést, a legsötétebb piros szín pedig azt, hogy az adott pixelen (minden pixel az vízszintes tengely mentén 0.072, a függőleges tengely mentén pedig 18.15 egységen vizsgálja a gyakoriságot) 50 a feltérképezett minimumok gyakorisága. A kettő közötti átmenetet különböző kék és sárga színek jelölik.

A 38 atomos rendszer density plotjához képest a 100 atomos rendszer mutat hasonlóságokat, illetve eltéréseket is. Vizsgáljuk meg a függőleges tengelyen 0-hoz közeli energiaszinteket. Míg a 38 atomos rendszer esetén nagyobb feltérképezetlen régiókat látunk, és nagyobb ugrások kellettek az alacsonyabb energiaszintekre történő betörésekhez, addig a 100 atomos rendszer esetén az energiaszintek közelebb vannak egymáshoz. Mivel a feltérképezett energiaszintek változnak az atomok számával, ezt a jelenséget érdemes az átlagos feltérképezett energiaszintekhez mérten vizsgálni.

Atomszám	Potenciális energia átlaga	Átlagos energiaszint távolság
20	2.546	0.483
30	2.782	0.032
38	4.736	0.129
40	4.722	0.085
50	5.926	0.014
60	6.234	0.012
70	7.569	0.007
80	6.61	0.004
90	8.374	0.002
100	9.606	0.001

4.5. táblázat: A 38 atomos rendszer esetén 1000 futásból, a többi esetben 100 futásból számolt átlagos algoritmus által feltérképezett potenciális energiaszint, illetve az átlagos feltérképezett energiaszint és a referenciaeredmény közötti energiatartomány alsó 40%-ában a sorba rendezett energiaszintek energiában mért átlagos távolsága.

Az atomszám növekedésével növekszik az algoritmus által feltérképezett energiaszintek átlaga, ezzel egyidejűleg csökken az átlag energiaszint és a referenciaeredmény közötti energiatartomány alsó 40%-ában a sorba rendezett energiaszintek energiában mért átlagos távolsága. Az átlagos távolság csökkenése miatt egyre kisebb ugrásokra van szüksége az algoritmusnak a referenciaeredmény közelében nagyobb atomszám esetén. Az átlagos távolságban a 20 és 30 atomos rendszerek között nagy ugrás látható, ez lehet az oka annak, hogy a 4.2-es táblázatban eltérő eredményt mutat a 30 és 38 atomos rendszerekhez képest. Ezen szimulációs eredmények alapján arra következtethetünk, hogy a referenciaeredményhez közeli energiaszintek távolsága és az algoritmus paraméterezése kapcsolatban állnak egymással. A kapcsolat jellegének meghatározásához, illetve a legjobb paraméterezés atomszám függvényében történő meghatározásához további szimulációkra van szükség, amihez 100 atom feletti rendszereket is érdemes lesz megvizsgálni.

5. fejezet

Konklúzió

Új megoldást mutattunk be a PSO algoritmus gyorsítására, mely véges differencia alapú gradiens becslést használ a részecskék sebességének változtatására, és nem igényel több függvénykiértékelést, mint az eredeti algoritmus. A módszer segítségével több optimalizációs tesztprobléma is megoldásra került. Általános tesztproblémák esetén a módszer jobban teljesített a kevés lokális szélsőértéket tartalmazó rendszerekre, illetve 5 változó alatt a szimulációk többségében mint a bővítést nem használó változat.

A Minima Hopping algoritmus hatékonysága 100 atomos rendszerig került tesztelésre. Hatékonysága jóval meghaladja a PSO algoritmusét Lennard-Jones feladatokra. Az irodalmitól eltérő paraméterezés segítségével közel 50%-os javulást sikerült elérnünk a számítási időben 38 atomos rendszer esetén és közel 20%-os javulást 100 atomos rendszerre. 38 atomos rendszerre az irodalmi megoldások átlagos futásideje 49.5 és 46.03 másodperc, még az új paraméterekkel 26.33 másodperc volt 1000 futás átlagát vizsgálva Intel Core i7 3630QM CPU-n párhuzamosítás nélkül, ami egy futtatás esetében nem tűnik sok időnek, viszont a terület kutatói nem csak a rendszerek energiaminimális állapotát keresik, hanem állapotok közötti átmeneteket is igyekeznek feltérképezni, mert ezen átmenetek tanulmányozása hozzásegítheti őket új anyagok előállításához [25], ami egy adott rendszer esetén sok szimulációt és összetett matematikai módszereket igényel. Sok futtatás esetén így jelentős az időnyereség. A szimulációk során láthatóvá vált, hogy a megfelelő paraméterezés függ az atomszámtól, és a globális szélsőérték közelében található energiaszintek távolságától. A 4.2-es táblázat alapján kirajzolódik egy trend arra vonatkozóan, hogy melyik paraméterezés a jobb az atomszám függvényében, viszont összevetve az adatokat a 38 atomos rendszert beállításonként 1000 futással vizsgáló 4.4-es táblázattal, látható, hogy a 4.2-es tartalmaz némi zajt, a 100 futás helyett érdemes lenne minden esetben 1000 futást vizsgálni több paraméterbeállítással, több rendszerre, viszont ahhoz egy PC-nél nagyobb hardware erőforrásra van szükség. A dolgozatban feltett kérdések megválaszolásában sikerült előrelépést elérni. Milyen paraméterek esetén lesz gyorsabb az algoritmus anélkül, hogy a módszer a hatékonyságából veszítene? Azt látjuk, hogy eléggé széles skálán mozognak a használható paraméterek a referenciaeredmény megtalálásának a szempontjából, az algoritmus szinte mindig megtalálta a referenciaeredményt, csak különböző idő alatt. Melyik a legjobb paraméter választás és hogyan függnek a paraméterek az aktuális rendszertől? A legjobb paraméterválasztás további kutatást igényel, viszont a paraméterek rendszertől történő függésében sikerült előre lépni.

Irodalomjegyzék

- [1] J. Kennedy, R. Eberhart: *Particle swarm optimization*, IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948, 1995.
- [2] J. Kennedy, R. Eberhart: *A new optimizer using particle swarm theory*, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43, 1995.
- [3] S. He, E. Prempan, Q. H. Wu: *An improved particle swarm optimizer for mechanical design optimization problems*, Engineering Optimization, 36: 5, 585 — 605, 2004.
- [4] C.R. Suribabu, T.R. Neelakantan: *Design of water distribution networks using particle swarm optimization*, Urban Water Journal, 3: 2, 111 — 120, 2006.
- [5] S. Vakili, M. S. Gadala : *Effectiveness and Efficiency of Particle Swarm Optimization Technique in Inverse Heat Conduction Analysis*, Numerical Heat Transfer, Part B: Fundamentals, 56: 2, 119 — 141, 2009.
- [6] M. Mologa, C. Smutnicki: *Test functions for optimization needs*, 2005.
- [7] E. Mendel , R. A. Krohling, M. Campos: *Swarm algorithms with chaotic jumps applied to noisy optimization problems*, 2011.
- [8] R.V. Rao, V.J. Savsani, D.P. Vakharia *Teaching-Learning-Based Optimization: An optimization method for continuous non-linear large scale problems*, 2012.
- [9] J. E. Jones, d A. E. Ingham, *Proc. R. Soc. A* 107, 636., 1925.
- [10] E. Fan: *Global Optimization Of Lennard-Jones Atomic Clusters*, M.Sc Thesis, McMaster University, 2002.
- [11] D. J. Wales, J. P. K. Doye: *Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms*, 1998.
- [12] T. Vinkó: *Globális optimalizálási módszerek továbbfejlesztése, tesztelése és alkalmazása atomklaszter feladatokra*, phd, Szegedi Tudományegyetem, 2006.
- [13] D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. Naumkin F. Calvo, J. Hernández-Rojas, T. F. Middleton: *The Cambridge Cluster Database*, URL <http://www-wales.ch.cam.ac.uk/CCD.html>
- [14] V. E. Bazterra, M. B. Ferraro, J. C. Facelli: *Modified genetic algorithm to model crystal structures. I. Benzene, naphthalene and anthracene*, J. Chem. Phys. 116, 5984, 2002.

-
- [15] A. R. Oganov, C. W. Glass: *Crystal structure prediction using ab initio evolutionary techniques: Principles and applications*, *J. Chem. Phys.* 124, 244704, 2006.
 - [16] S. E. Schönborn, S. Goedecker, S. Roy, A. R. Oganov *The performance of minima hopping and evolutionary algorithms for cluster structure prediction*, *J. Chem. Phys.* 130, 144108, 2009.
 - [17] S. Goedecker: *Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems*, *The Journal of Chemical Physics* 120, 9911, 2004.
 - [18] M. Grasselli, D. Pelinovsky: *Numerical Mathematics*, Jones And Bartlett Publishers Canada, ISBN-13: 978-0763737672, 2008.
 - [19] M. M. Noel, T. C. Jannett: *Simulation of a New Hybrid Particle Swarm Optimization Algorithm*, 2004. *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory*, 150 - 153, ISSN : 0094-2898, 2004.
 - [20] D. Barla-Szabó: *A Study of Gradient Based Particle Swarm Optimisers*, Msc Thesis, University of Pretoria, Pretoria, 2010.
 - [21] J.J.Liang, P.N.Suganthan: *Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search*, *The 2005 IEEE Congress on Evolutionary Computation*, 522 - 528 Vol.1, 2005
 - [22] Cs. Barcsák, K. Jármai: *Benchmark for testing evolutionary algorithms*, 10th World Congress on Structural and Multidisciplinary Optimization, May19-24, 2013, Orlando, Florida, USA (URL <http://www2.mae.ufl.edu/mdo/Papers/5124.pdf>), 2013.
 - [23] S. Roy, S. Goedecker, W. Hellmann: *Bell-Evans-Polanyi principle for molecular dynamics trajectories and its implications for global optimization*, *Physical Review E* 77, 056707, 2008.
 - [24] M. Amsler, S.Goedecker: *Crystal structure prediction using the minima hopping method*, *The Journal of Chemical Physics* 133, 224104, 2010.
 - [25] B. Schaefer, S. Mohr, M. Amsler, S.Goedecker: *Minima hopping guided path search: An efficient method for finding complex chemical reaction pathways*, *The Journal of Chemical Physics* 140, 214102, 2014.