The Structural Power of Reconfigurable Circuits in the Amoebot model

Béres Gábor Kristóf (P2J97V), Werner Bendegúz (A3B7DU), Tóth Botond (MQH41V)

Absztrakt

Az „The Structural Power of Reconfigurable Circuits in the Amoebot model” című tanulmány egy olyan aktív programozható anyagot mutat be, ahol az egységek – az úgynevezett amőbotok – egy végtelen háromszög rácson helyezkednek el, képesek mozogni, kommunikálni és számításokat végezni. A kutatás középpontjában az áll, hogy miként optimalizálható az amőbotok kollektív viselkedése a számítási és energiaigény csökkentése érdekében. A modellhez bevezetett átprogramozható áramköri kiterjesztés révén az amőbotok képesek gyors és hatékony jelátvitelre, amely lehetővé teszi bonyolult algoritmusok, pl globális maximum meghatározása, csontváz, spanning tree létrehozása és szimmetriaellenőrzés hatékony végrehajtását. A projekt keretében részletes algoritmusokat dolgoztunk ki több kulcsfontosságú probléma megoldására: stripe azonosítás, globális maximum keresés, ezket az algoritmusokat vizuálisan szimuláltuk, illetve ezeknek az eredményeit és futási teljesítményét elemeztük nagy mennyiségű amőbot struktúrára. Illetve ezeket az amőbot struktúrákat átalakítottuk egy algoritmus segítségével jégcsap szerkezetbe, és az eredeti és jégcsap szerkezetre futtattuk az algoritmusokat és összehasonlítottuk a futási eredményeket. Az algoritmusok mindegyike polilogaritmikus időbonyolultságú, jelentős teljesítményjavulást biztosítanak a hagyományos megközelítésekhez képest. A dokumentációban bemutatott elméleti és algoritmikus eredmények hozzájárulhatnak a programozható anyagok jövőbeli alkalmazásaihoz, mint például intelligens robotikus rendszerek, nanoszintű feladatok automatizálása.

Bevezetés

A programozható anyag egy modern koncepció, amely az elmúlt időszakban jelentős kutatási érdeklődésre tett szert, különösen az orvosi technológia, nanotechnológia és környezetvédelem területén. A programozható anyagok olyan anyagról, amelyek képesek kollektíven komplex viselkedést tanúsítani, például alakjukat változtatni és tulajdonságait előre programozott módon megváltoztatni, ezáltal különféle feladatok elvégzésére alkalmas alakzatokat és szerkezeteket hozhatnak létre. A programozható anyagok alapvető alkalmazási területei közé tartoznak a nanoszintű műtétek, szennyező anyagok eltávolítása a környezetből. A programozható anyagok lényege tehát az alakzatok dinamikus átalakítása.

Programozható anyagok rendszerezése: passzív és aktív rendszerek

A programozható anyag modellek különböző típusokba sorolhatók attól függően, hogy mennyire képesek az anyag elemei önállóan mozogni és számításokat végezni. Az úgynevezett passzív rendszerekben az anyag elemei, csak külső hatásokra reagálnak, például fény, áram, vagy mágneses mezők hatására mozdulnak meg. Ezek az elemek nem rendelkeznek számítási képességgel, és kizárólag külső behatás segítségével mozhathatók (pl. elektromos mező, fény impulzus vagy robotkarok).

Az aktív rendszerek esetében az anyag egyes részecskéi, robotok, képesek az önálló mozgásra, tudnak kommunikálni egymással és tudnak számítási műveleteket végrehajtani, például önállóan döntéseket hozhatnak a következő lépéseikről. Ezek az aktív modellek, különösen jól alkalmazhatók olyan rendszerekben, ahol az anyagok teljesen önálló átalakulást végeznek.

A projektben vizsgált Amőbamodell egy ilyen aktív modell, amely apró robotikus egységekből (amőbotokból) áll.

Ezek egy végtelen háromszög rácson helyezkednek el, van egy közös iránytűjük, ami megadja merre néznek az amőbotok, illetve összefüggő konfigurációban helyezkednek el az egyes amőbotok. Képesek mozogni tágulás és összehúzódás révén globális koordináció nélkül, ahol különféle alakzatokat, például hexagonokat vagy háromszögeket alkothatnak.

Viszont ezeknek az aktív modelleknek a hátránya, hogy sokszor magas számítási és energiaigényük van, mivel minden részecske képes önállóan mozogni és számításokat végrehajtani, és ez akadályozhatja a struktúrák hatékony átalakítását.

Ennek a projektnek a célja, a kollektív viselkedésnek a vizsgálata, és optimalizálása, a számítási igény csökkentése érdekében (polilogaritmikus algoritmus futási idők elérése).

Kapcsolódó munkák, modell kiterjesztés

A “The Structural Power of Reconfigurable Circuits in the Amoebot model” alapját korábbi kutatások adták, amelyek meghatározták az amőbot programozható anyag modell struktúráját és alap számítási és mozgási képességeit.

Ez a projekt ezeket a modelleket bővítette az áramköri kiterjesztéssel, aminek a lényege, hogy az egyes amőbotok közötti éleket lecserélték külső kapcsolatokra, minden szomszédos amőbot között legalább egy ilyen kapcsolatnak kell létezni, de lehet több is. Ezeknek a végpontjait felosztották diszjunkt halmazokra, ezáltal kialalkulnak áramkörök, amelyeken képesek az amőbotok primitív kommunikációra egymással (beep).

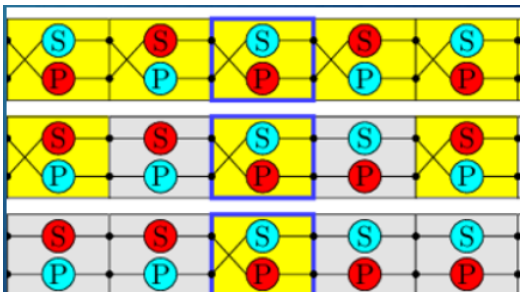
Ezzel lehetővé tették az azonnali jelátvitelt, aminek segítségével képesek különböző akciókat végrehajtani

PASC algoritmus működése

A PASC algoritmus célja, hogy amőbotokból álló lánc struktúrákon azonosítókat számoljon ki az egyes amőbotokhoz. Az algoritmus a következő fő lépésekből áll:

- **Referencia amőbot meghatározása**: Az algoritmus vezetőválasztás vagy valamilyen más módszer segítségével meghatározza a referencia amőbotot. Az algoritmus innen fog indulni, és minden amőbot a referencia amőbothoz tartozó relatív pozíciója alapján kapja majd az azonosítókat.
- **Elsődleges és másodlagos áramkör létrehozása**: Meghatározunk 2 áramkört, ami a lánc összes elemét összeköti, a 2 áramkör nincs összekötve egymással, és ezeknek az áramköröknek a segítségével meghatározunk egy elsődleges és másodlagos partícióhalmazt. A partícióhalmaz elemeit összekötjük egymással következőképpen:
  - ➢ Aktív elsődleges elem elődje másodlagos elem
  - ➢ Aktív másodlagos elem elődje elsődleges
  - ➢ Passzív elsődleges elem elődje elsődleges
  - ➢ Passzív másodlagos elem elődje másodlagos

  Az algoritmus elején minden elem aktív



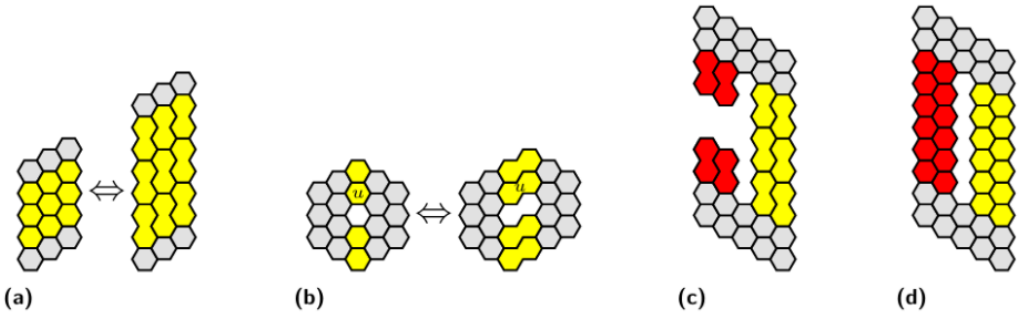
- **Azonosítók kiosztása**: A referencia amőbot aktiválja az elsődleges áramkört (küld egy primitív jelzést az áramkörön). Aki a másodlagos áramkörén kapta meg a jelzést, az a 2. körben a másodlagos áramkörén fog jelezni, majd passzív lesz (ezáltal átkonfigurálja a kapcsolatait). Minden amőbot 1 körben csak az egyik áramkörén fog jelzést kapni. Az azonosítók kiosztása a jelzések segítségével történik, minden jelzés egy bit:
  - ➢ Ha az amőbot az elsődleges áramkörén kapott jelzést: 0-ás bitet jegyez fel
  - ➢ Ha az amőbot a másodlagos áramkörén kapott jelzést: 1-es bitet jelez fel
  - ➢ Minden bitet visszafele kell jegyezni, és kettős komplemens számábrázolást használunk, ezzel lehetővé téve negatív azonosítók kiosztását

  Az algoritmus addig fog futni, amíg a 2. kör csendes nem marad, ha vége van, akkor megkaptuk a referencia amőbothoz képest az egyes amőbotok azonosítóit, amelyet majd további problémák megoldására fel lehet használni.

Stripe problem

A Stripe probléma lényege, hogy egy adott tengely mentént határozzuk mely amőbotok tartoznak az adott csíkba.

Ez elengedhetetlen a mozgások során az ütközések elkerülésére, illetve a gyors alakváltáshoz.



A problémát megoldja a PASC algoritmus, meg kell határozni egy irányt és a referencia amőbotot, hogy melyik node-ról indoljon, és egy amőbot struktúrában meg fogja határozni az adott iránynak és referencia amőbotnak megfelelő csíkot.

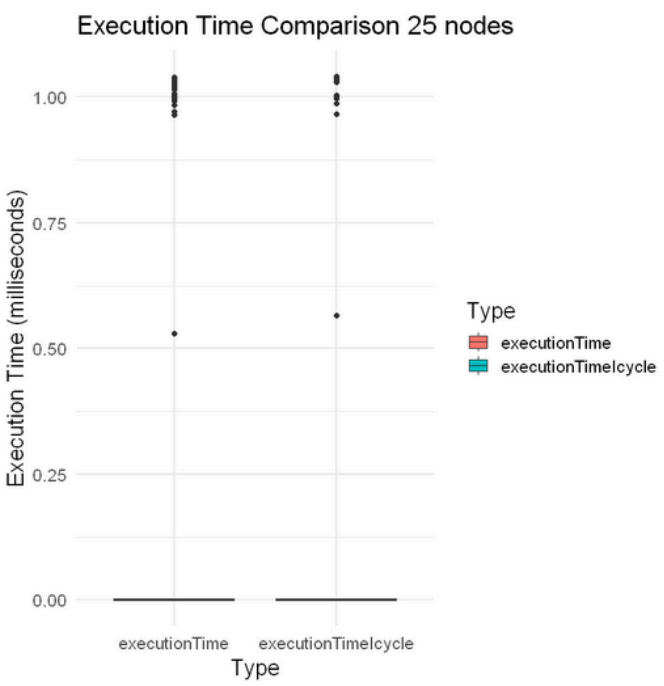
Eredmények

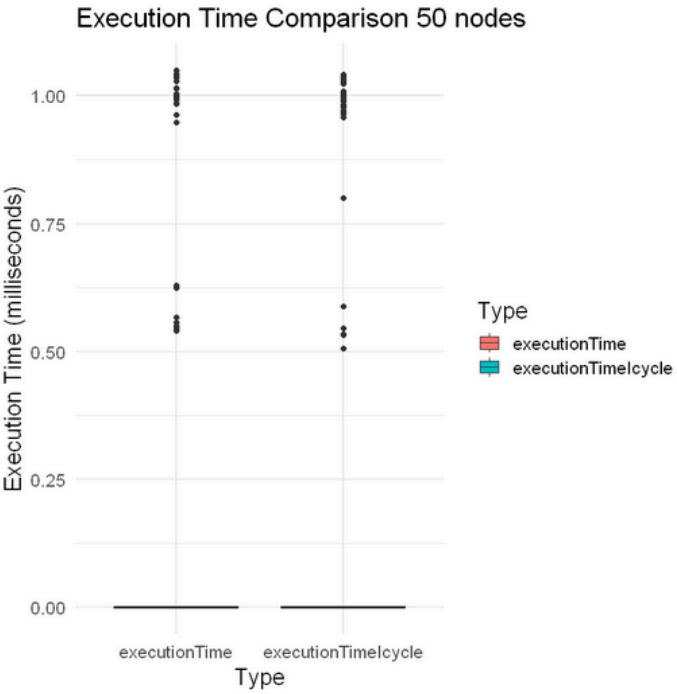
A stripe algoritmus futási ideje O(log n), ami polilogaritmikus futási idejű.

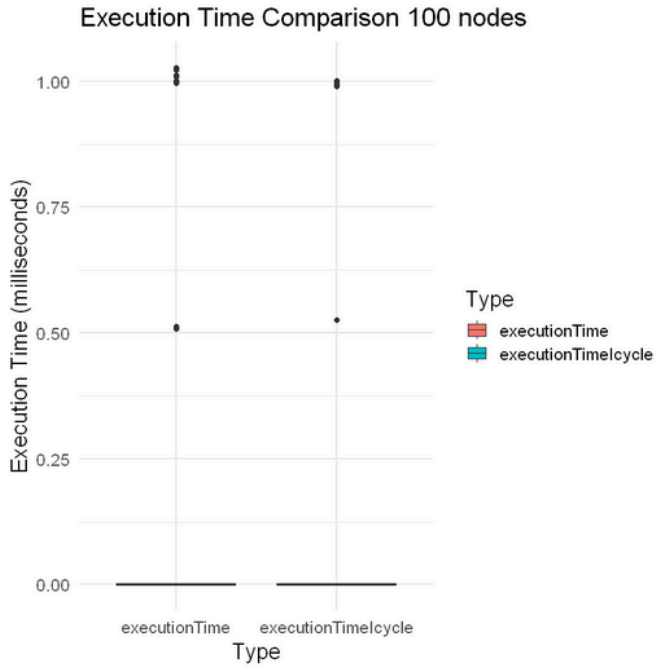
A vizuális szimuláció nem használható nagy mennyiségű adat vizsgálatára, illetve futási idők mérésére, ezért egy programmal a vizuális elemek nélkül futtatuk az algoritmust nagy mennyiségű (~2000) különböző méretű amőbot struktúrákra (25, 50, illetve 100 node) és vizsgáltuk az algoritmus paramétereit (futási idő, lépésszám).

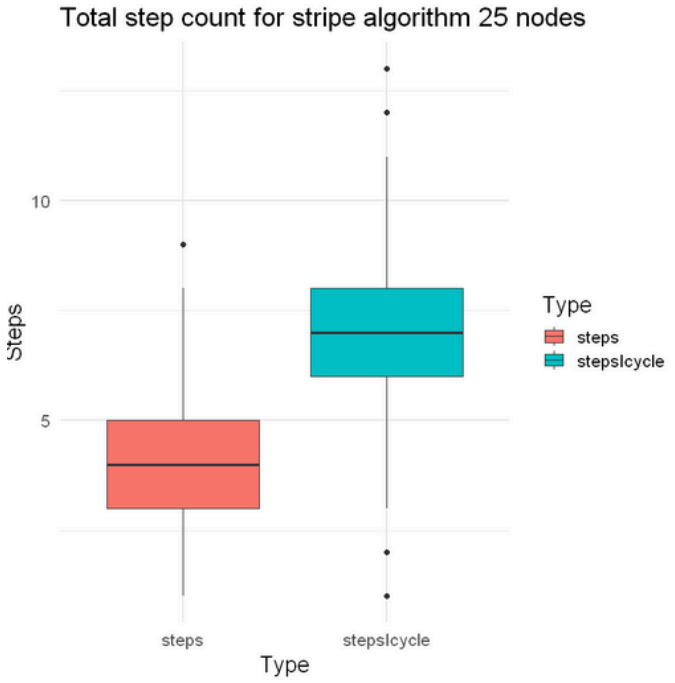
Illetve a generált amőbotstruktúrákat átalakítottuk jégcsap struktúrákba, azokra is lefuttattuk az algoritmust, és összehasonlítottuk az eredeti struktúrával, hogyan változtak az algoritmus paraméterei.

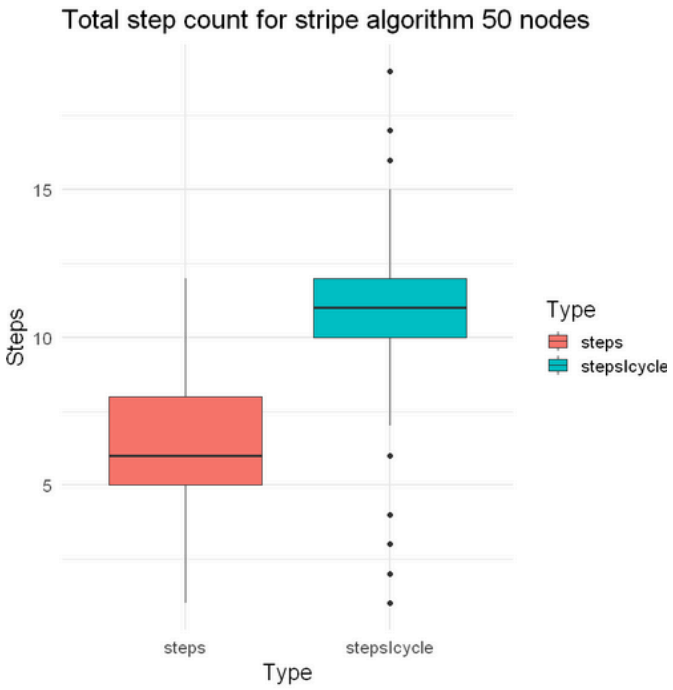
A futási eredményeket csv fájlokba írtuk ki, és az adatokat R-ben feldolgoztuk és elemeztük:

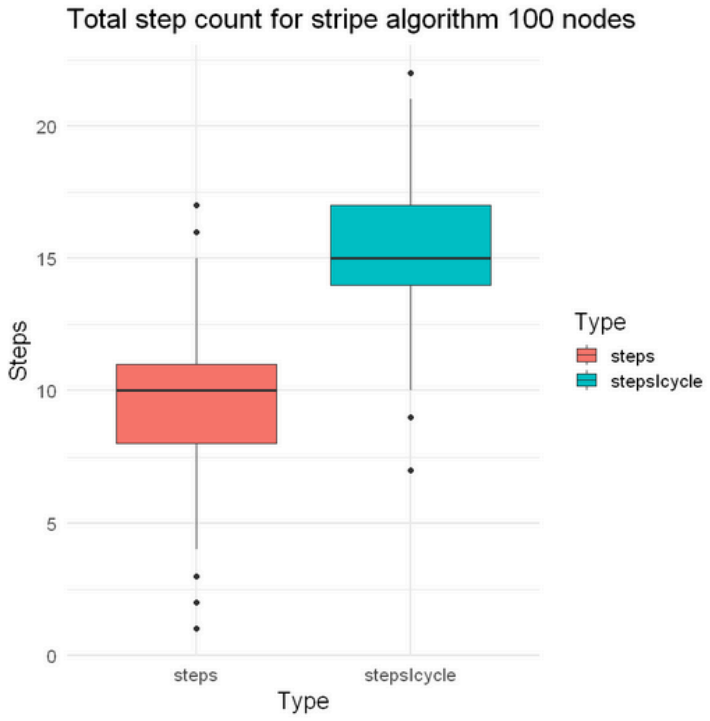

Execution Time Comparison 25 nodes
Execution Time (milliseconds)
1.00
0.75
0.50
0.25
0.00
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle


Execution Time Comparison 50 nodes
Execution Time (milliseconds)
1.00
0.75
0.50
0.25
0.00
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle


Execution Time Comparison 100 nodes
Execution Time (milliseconds)
1.00
0.75
0.50
0.25
0.00
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle


Total step count for stripe algorithm 25 nodes
Steps
10
5
steps
stepsIcycle
Type
Type
steps
stepsIcycle


Total step count for stripe algorithm 50 nodes
Steps
15
10
5
steps
stepsIcycle
Type
Type
steps
stepsIcycle


Total step count for stripe algorithm 100 nodes
Steps
20
15
10
5
0
steps
stepsIcycle
Type
Type
steps
stepsIcycle

Globális maximum probléma

A globális maximum problémának a célja, hogy meghatározzuk egy adott amőbot struktúra legszélső elemeit egy adott iránynak megfelelően.

Ez a maximum meghatározható, ha felosztjuk az amőbotokat stripe-okra az adott tengely mentén, és a tengelyekre lefuttatjuk a PASC algoritmust, és a kapott azonosítókat kiértékeljük.

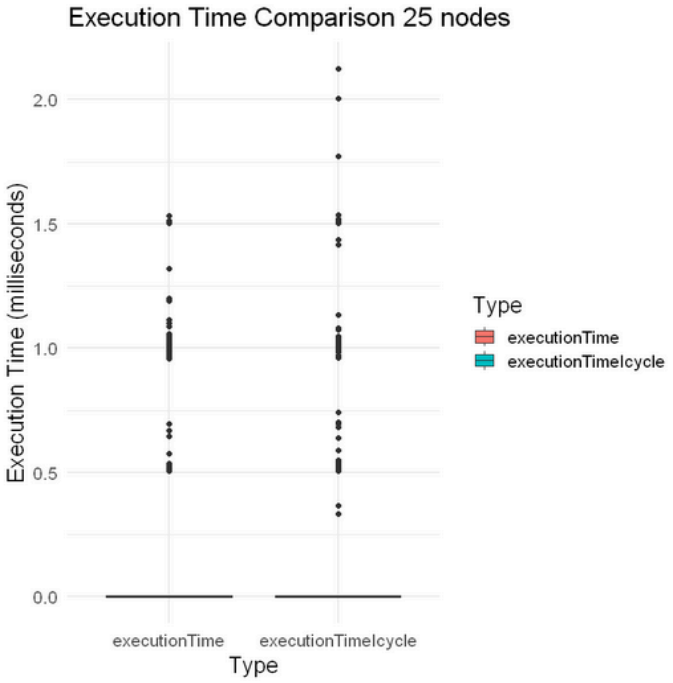
Eredmények

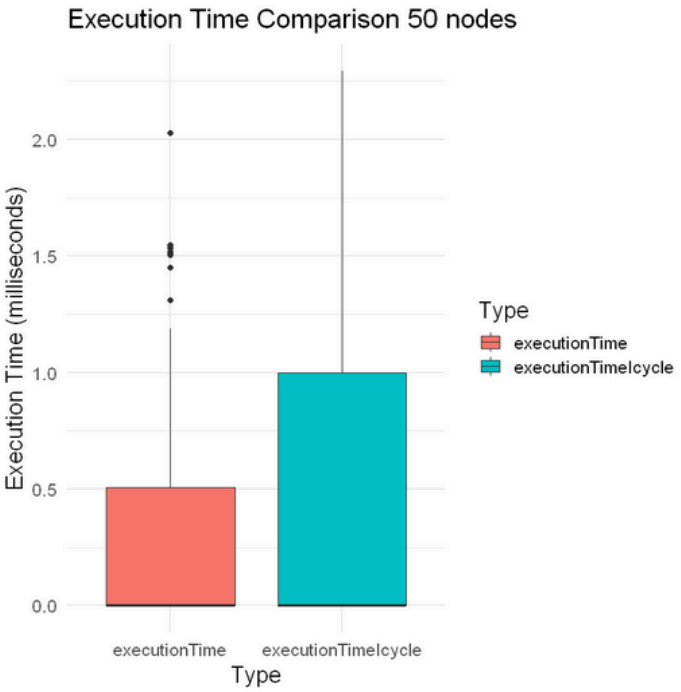
A globális maximum algoritmus futási ideje O(log2n), ami polilogaritmikus futási idejű.

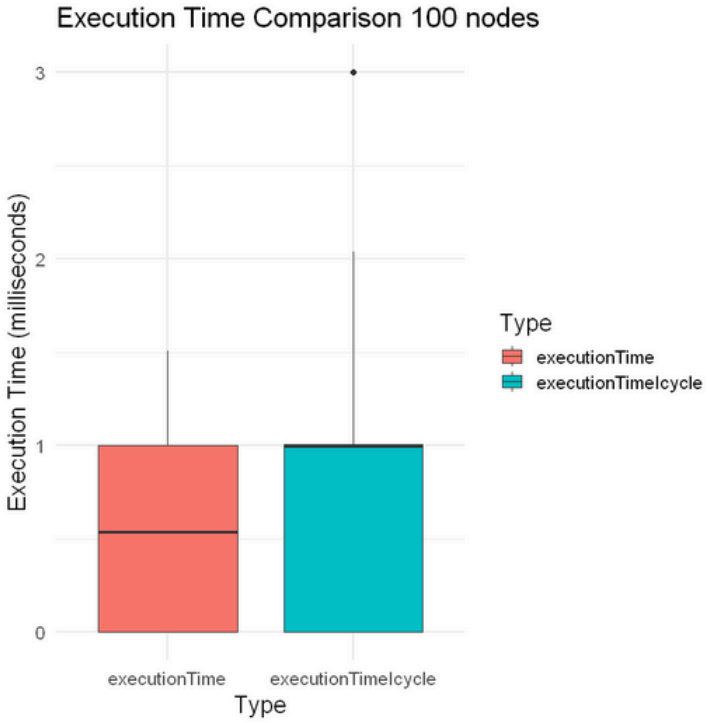
A vizuális szimuláció nem használható nagy mennyiségű adat vizsgálatára, illetve futási idők mérésére, ezért egy programmal a vizuális elemek nélkül futtatuk az algoritmust nagy mennyiségű (~2000) különböző méretű amőbot struktúrákra (25, 50, illetve 100 node) és vizsgáltuk az algoritmus paramétereit (futási idő, PASC iterációk számal, leghosszabb stripe-ra lépésszám, összes lépésszám).

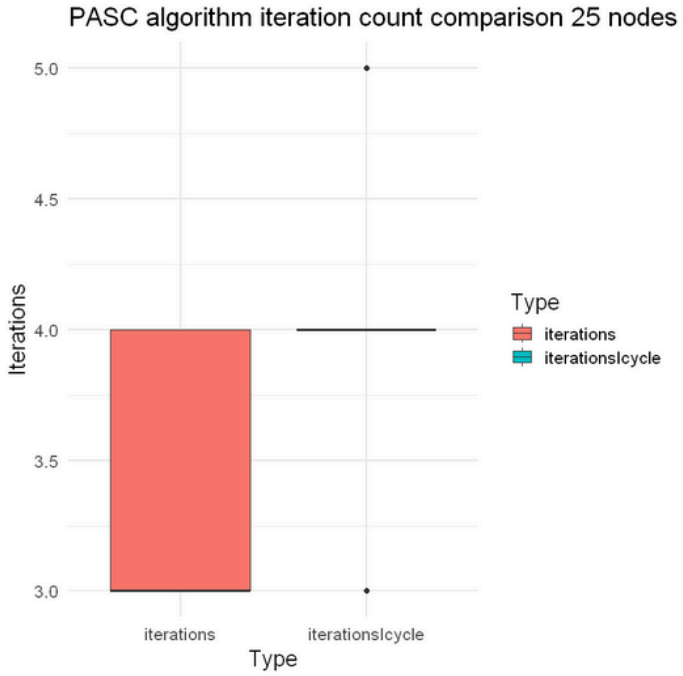
Illetve ezt a ~2000 generált amőbotstruktúrát átalakítottuk jégcsap struktúrákba, azokra is lefuttattuk az algoritmust, és összehasonlítottuk az eredeti struktúrával, hogyan változtak az algoritmus paraméterei.

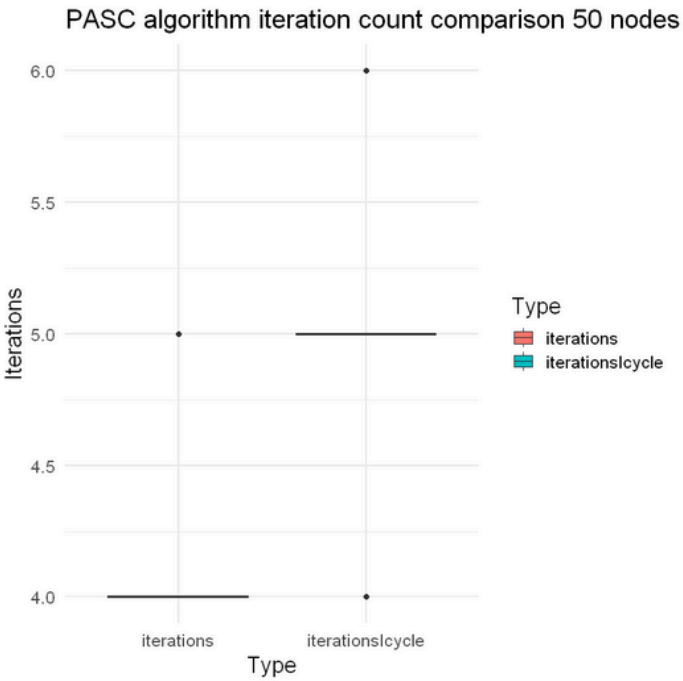
A futási eredményeket csv fájlokba írtuk ki, és a nagy mennyiségű adatot R-ben feldolgoztuk és elemeztük:

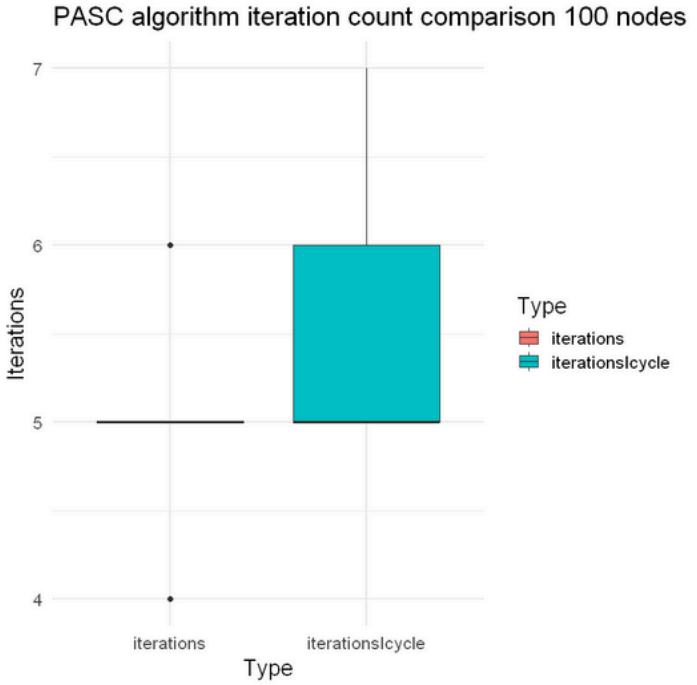

Execution Time Comparison 25 nodes
Execution Time (milliseconds)
2.0
1.5
1.0
0.5
0.0
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle

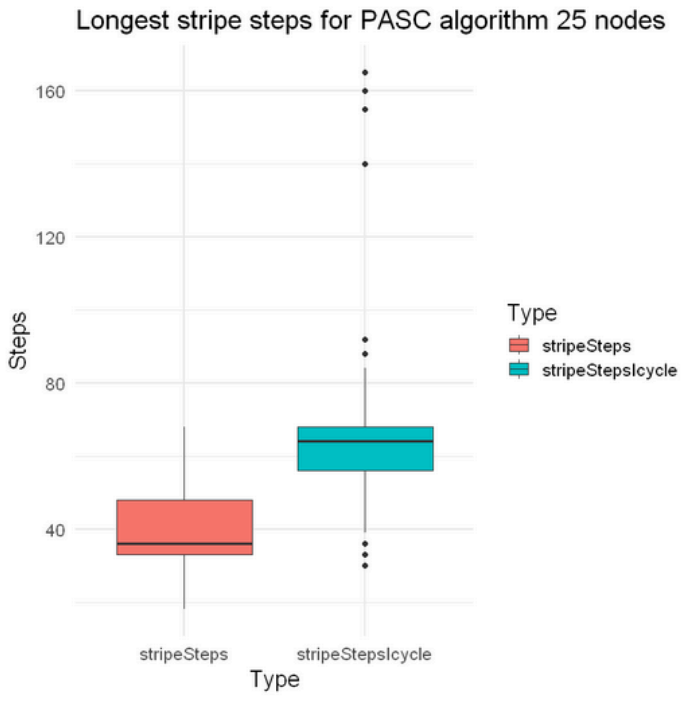

Execution Time Comparison 50 nodes
Execution Time (milliseconds)
2.0
1.5
1.0
0.5
0.0
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle

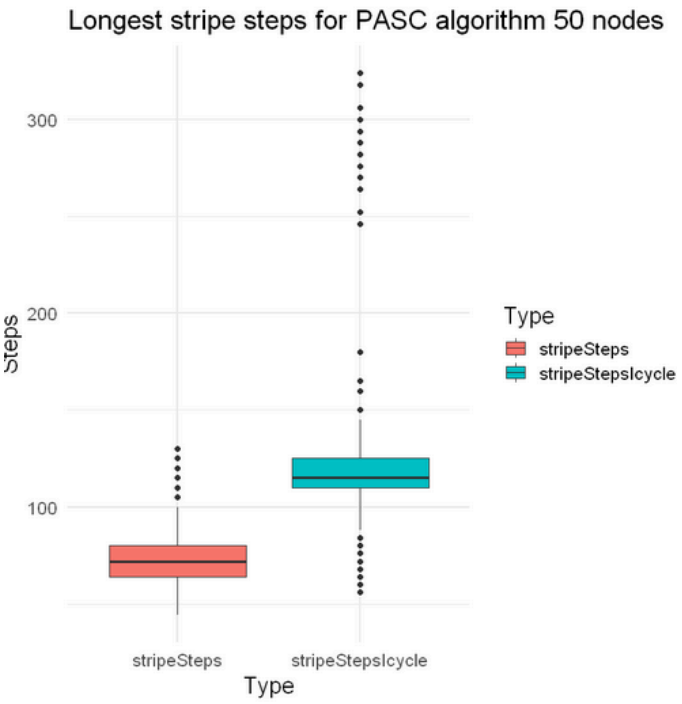

Execution Time Comparison 100 nodes
Execution Time (milliseconds)
3
2
1
0
executionTime
executionTimeIcycle
Type
Type
executionTime
executionTimeIcycle

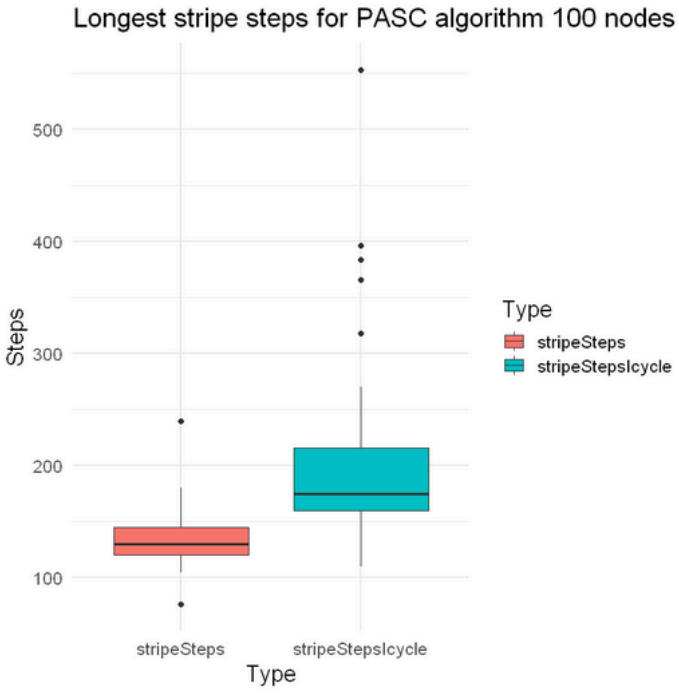

PASC algorithm iteration count comparison 25 nodes
5.0
4.5
4.0
3.5
3.0
Iterations
iterations
iterationsIcycle
Type
Type
iterations
iterationsIcycle

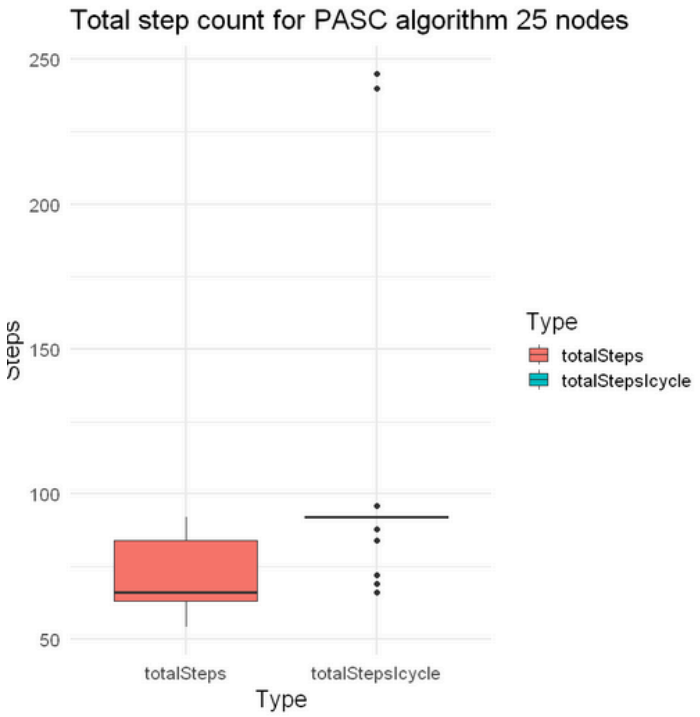

PASC algorithm iteration count comparison 50 nodes
6.0
5.5
5.0
4.5
4.0
Iterations
iterations
iterationsIcycle
Type
Type
iterations
iterationsIcycle

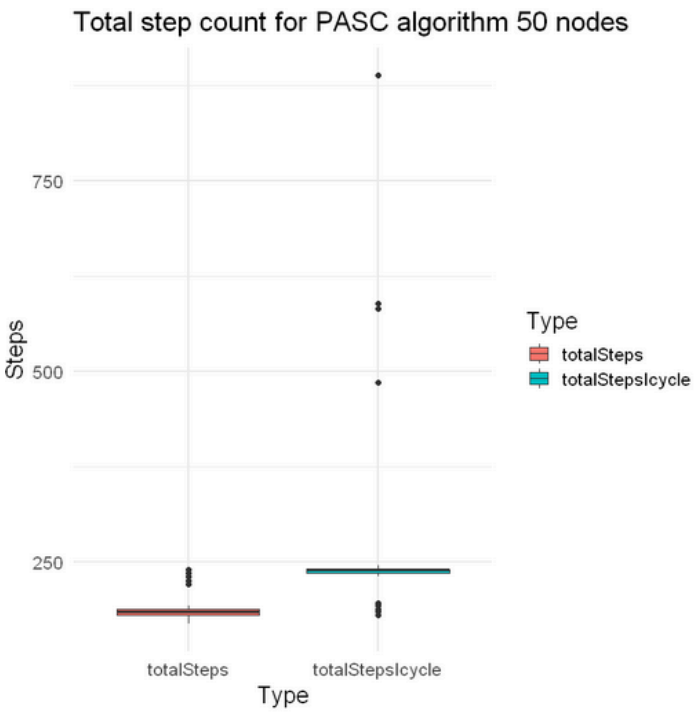

PASC algorithm iteration count comparison 100 nodes
7
6
5
4
Iterations
iterations
iterationsIcycle
Type
Type
iterations
iterationsIcycle

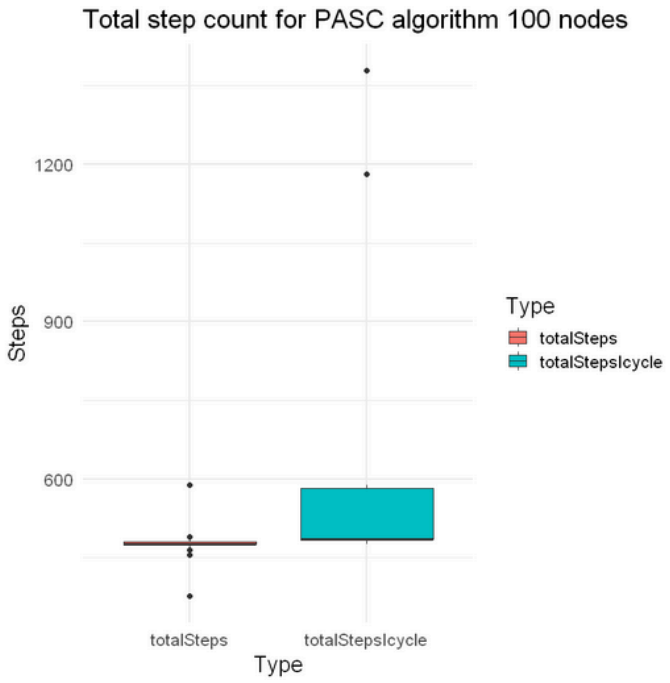

Longest stripe steps for PASC algorithm 25 nodes
160
120
80
40
Steps
stripeSteps
stripeStepsIcycle
Type
Type
stripeSteps
stripeStepsIcycle


Longest stripe steps for PASC algorithm 50 nodes
300
200
100
Steps
stripeSteps
stripeStepsIcycle
Type
Type
stripeSteps
stripeStepsIcycle


Longest stripe steps for PASC algorithm 100 nodes
500
400
300
200
100
Steps
stripeSteps
stripeStepsIcycle
Type
Type
stripeSteps
stripeStepsIcycle


Total step count for PASC algorithm 25 nodes
250
200
150
100
50
Steps
totalSteps
totalStepsIcycle
Type
Type
totalSteps
totalStepsIcycle


Total step count for PASC algorithm 50 nodes
750
500
250
Steps
totalSteps
totalStepsIcycle
Type
Type
totalSteps
totalStepsIcycle

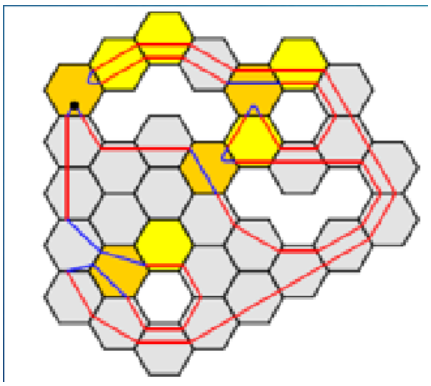

Total step count for PASC algorithm 100 nodes
1200
900
600
Steps
totalSteps
totalStepsIcycle
Type
Type
totalSteps
totalStepsIcycle

Csontváz probléma

A csontváz problémának a célja, hogy találjuk meg egy olyan ciklust, amely az összes határon lévő amőbotot tartalmazza.

Ezt a csontvázat egy algoritmus segítségével kell kialakítani, amihez meg kell határozni, hogy milyen irányban járjuk körbe az amőba struktúrát, és egy előjelet, ami megmondja, hogyan kössük majd összse a nem kapcsolódó határ amőbot köröket.

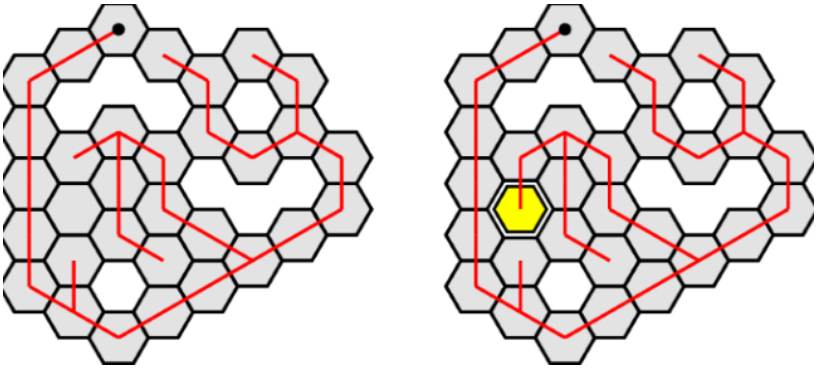
- Először meghatározzuk a globális maximumot a kiválasztott iránynak megfelelően, a globális maximum lesz a csontváz kiindulási pontja
- Meghatározzuk az elsődleges csontvázat, a kiindulási pontból az adott iránynak megfelelően, és meghatározzuk a többi határ amőbot kört
- A határ amőbotokat tartalmazó köröket összekötjük, és egyesítjük egyetlen csontvázzá egy jól meghatározott útvonal segítségével



Spanning tree létrehozása

Ha létre akarunk hozni egy spanning tree-t, egy amőbot struktúrához, azt az előző problémából kapott csontváz segítségével megtehetjük.

- Először is a csontváz útvonalban (amiben egy csúcs többször is előfordulhat), a PASC algoritmus segítségével meghatározzuk minden csúcs első előfordulását, és ezeket megtartva létrehozunk egy fát
- Ahhoz hogy spanning tree-t kapjunk, a fához hozzá kell adni a belső node-okat: Minden belső node-hoz hozzáadunk egy élt az északi szomszédjától (ami mindig létezik, mivel belső csúcsokra végezzük el a műveletet), ezáltal minden belső csúcsba is pontosan 1 él fog menni
- Ezzel megkaptuk a spanning tree-t, amit majd felhasználhatunk szimmetria vizsgálatokhoz



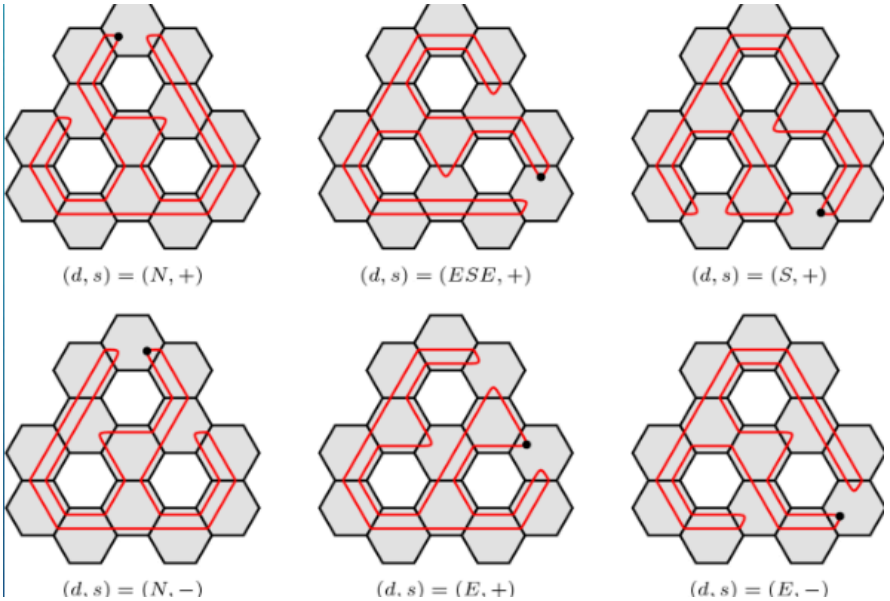
Szimmetria detektálás

Egy amőbotokból álló struktúra szimmetriái a különböző irányú és előjelű kanonikus csontvázak összehasonlításaival detektálható:

- minden (d, s) kanonikus csontvázat egy egyedi bit sorozattá alakítunk
- ► bitek összehasonlítása csontvázak helyett → polilogaritmikus futási idő: O(log^5 n)

Egy amőbot struktúrára a következő típusú szimmetriák értelmezhetőek:

- Északi irányú tengelyre szimmetrikus, ha az (N,+) és (N, -) csontvázak szimmetrikusak
- Keleti irányú tengelyre szimmetrikus, ha az (E, +) és (E, -) csontvázak szimmetrikusak
- 2-fold szimmetrikus (180°), ha az (N, +) és (S, +) csontvázak szimmetrikusak
- 3-fold szimmetrikus (120°), ha az (N, +) és (ESE, +) csontvázak szimmetrikusak
- 6-fold szimmetrikus (60°), ha 2 és 3-fold szimmetrikus



További kutatási lehetőségek

A tanulmány eredményei alapján több jövőbeli alkalmazási módszer kidolgozása lehetséges lenne:

Az elméletben meghatározott algoritmusok segítségével, amik mind polilogarimtikus futási idejűek, a stripe-ok segítségével nagyon jó futási idejű alakváltoztatásokat el lehetne érni, illetve a csontvázak létrehozásával és a polilogaritmikus szimmetriadetektálással az amőbot struktúrák monitorozását hatékonyan el lehetne végezni.

Ez egy jó megoldást nyújtana az aktív számítási rendszerek magas számítási teljesítményére és energiaigényére.