Efficient Shape Formation by 3D Hybrid Programmable Matter

HATÉKONY ALAKFORMÁLÁS 3D HIBRID PROGRAMOZHATÓ ANYAGGAL.

KÉSZÍTETTE: BÉRES GÁBOR KRISTÓF, PAULICSEK ÁDÁM,
TÓTH BOTOND

Motiváció

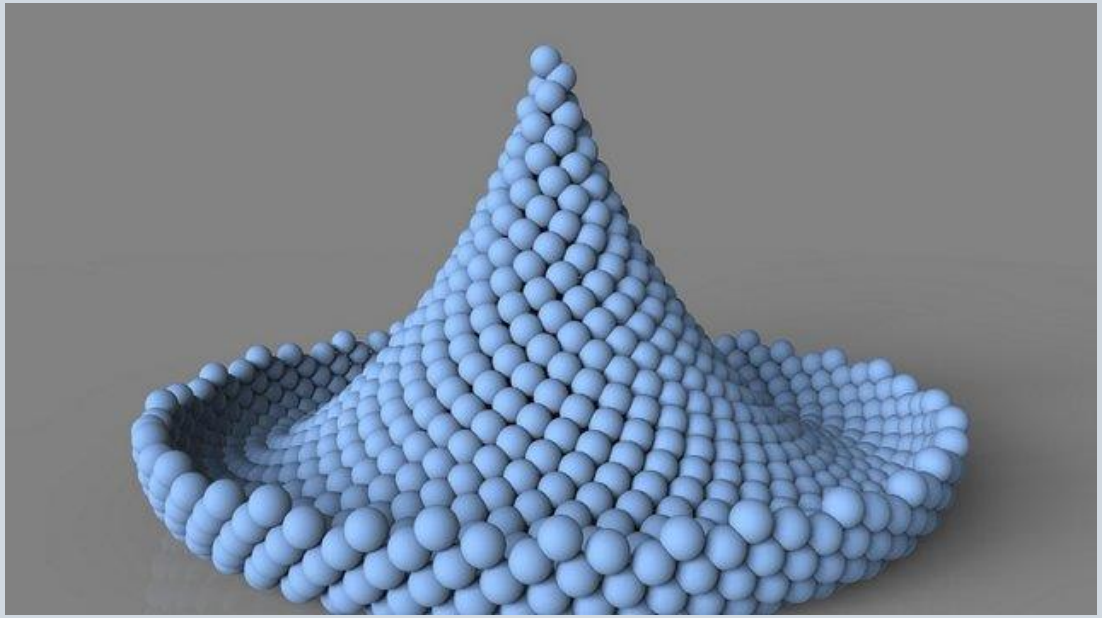
Programozható anyagok megjelenése

Tulajdonságok megváltoztatása

- Alakváltozás
- Színváltozás
- Szerkezeti változások

Irányítás

- Külső behatás
- programozott vezérlés



Programozható anyagok felépítése

Apró egységek

Együttműködés

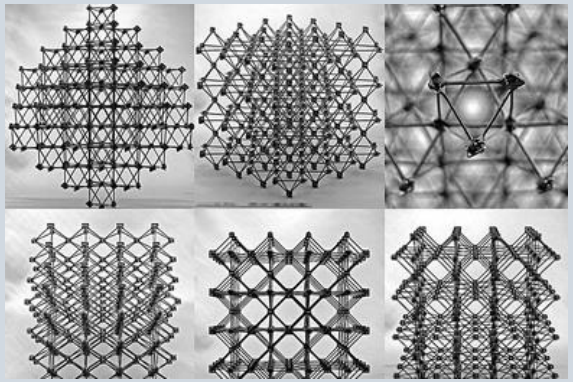
- Modularitás
- Adaptív viselkedés

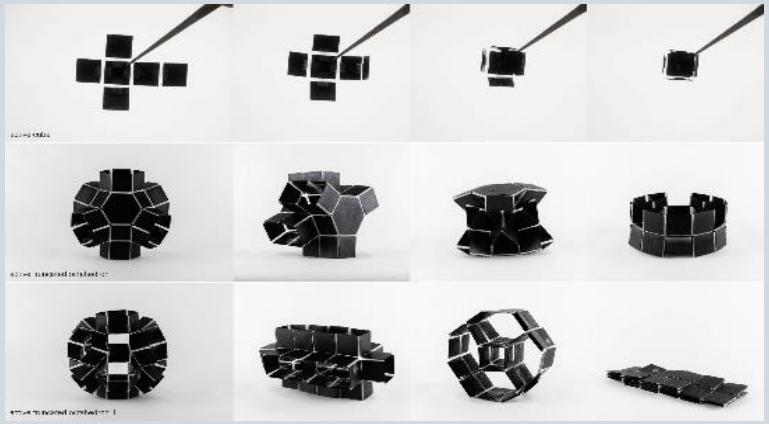
Ez a felépítés alapvetően határozza meg a programozható anyagok sokoldalú alkalmazhatóságát, lehetővé téve különböző struktúrák létrehozását és alakítását, akár valós időben is.

Aktív és passzív rendszerek

- ***hibrid modell***

Csempék és ügynök/aktor





Probléma és megoldás

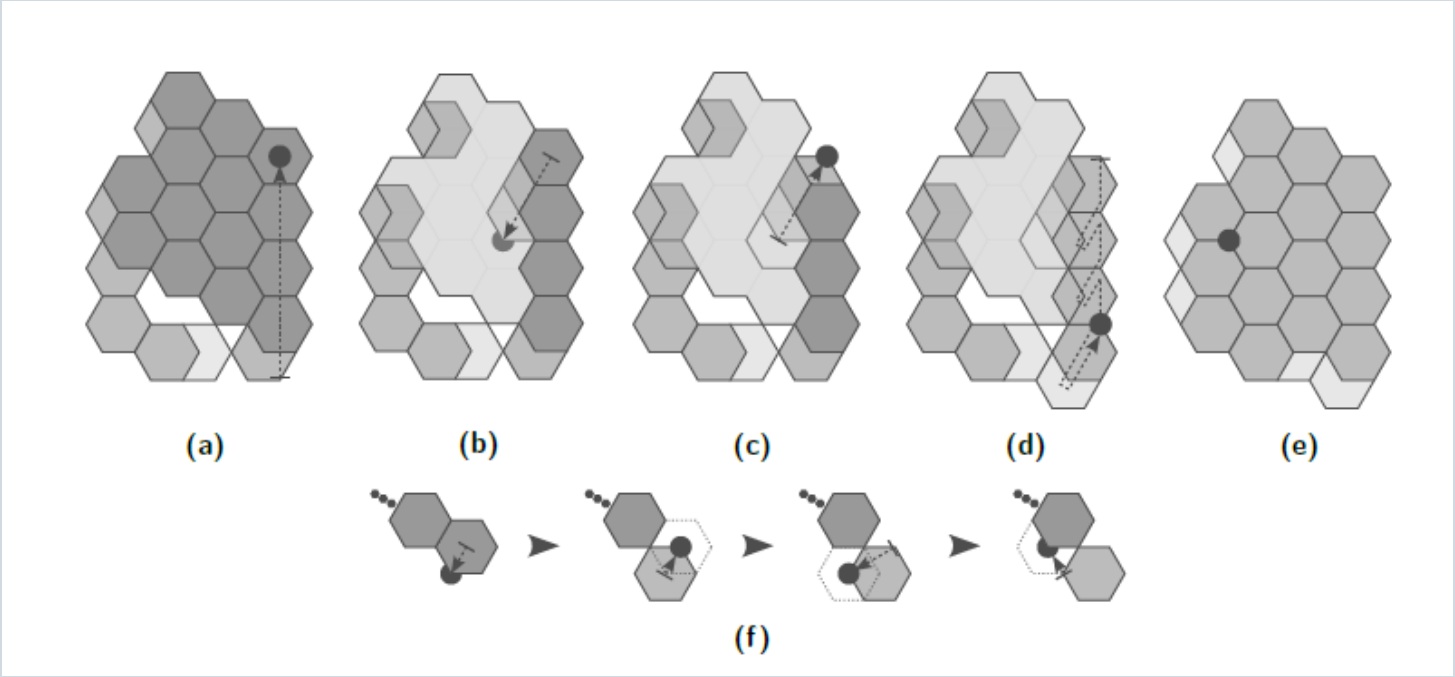
Formaalakítási problémák

Ügynök és csempék kapcsolata

Csempék manipulálása

Probléma

- Csempék mozgatása
- Hatékonyság növelés
- Formaalakítás

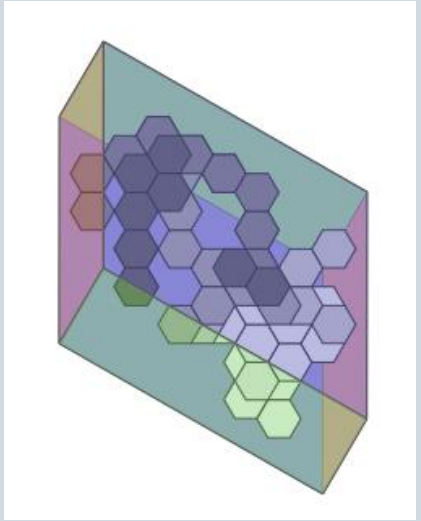


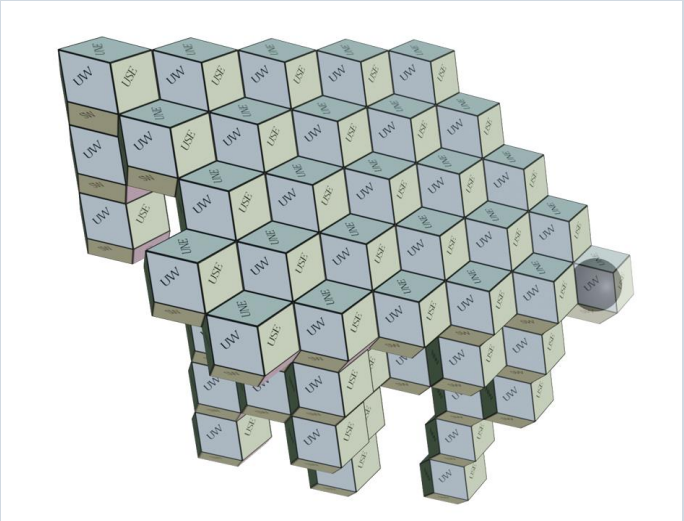
Jégcsap alakzat

Mozgatás a kapcsolat megszakítása nélkül

Cél: „jégcsap” struktúra

Előnyei: egyszerűsíti az alakformálást, kisebb átmérő, több mozgatható csempe -> ezek javítják az ügynök mozgásterét és hatékonyságát





Nanoszintű feladatok

Teher szállítása (gyógyszeradagolás, molekulák szállítása),

Kommunikáció elsősegítése (jelek fogadása és küldése egymás között vagy a környezetkkel)

Membránok felszínén való navigálás (nanorobotok képesek a sejtek vagy más biológiai struktúrák felületén mozogni)

Útvonalkeresés (ezek az egységek képesek lehetnek megtalálni a legjobb utat egy adott célhoz)

Felhasználás, konkrét lehetséges alkalmazások

Orvostudomány

- Gyógyszeradagolás
- Testbe juttatás

Környezetvédelem

- Nanoszűrők

Építőipar

- Sérülésjavítás

Elektronikai eszközök

- Rugalmasság (újraszerveződés feladattól függően)
- Hatékonyság növelés

Modell és probléma

Modell I.

Képzeljük el:
Az azonos méretű gömbök szoros elrendezését egy végtelen arccal középpontozott köbös rácson. Ez egy szabályos struktúra, ahol a gömbök egymáshoz közel, meghatározott mintában helyezkednek el.

Gráf létrehozása (G = (V, E)):

Csúcsok (V): A gömbök középpontjai.

Élek (E): Azok a kapcsolatok, ahol két gömb érintkezik egymással.

Beágyazás az R^3 térbe:
A gráfot ágyazzuk úgy be az R^3 térbe, hogy minden él azonos hosszúságú legyen. A triviális beágyazás esetében az élhossz megegyezik a gömbök sugarával, így a gömbök szomszédai közötti távolság is egyenlő.

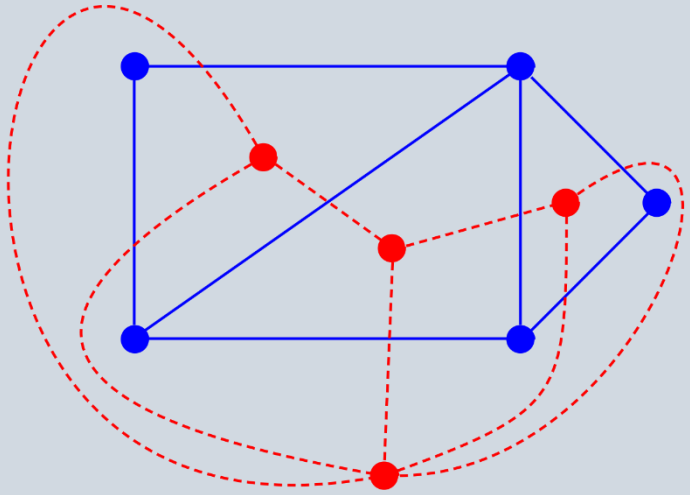
Aktív ügynök r:
Az ügynök, amelyet vizsgálunk, korlátozott érzékelési és számítási képességekkel rendelkezik a gráfban (G).

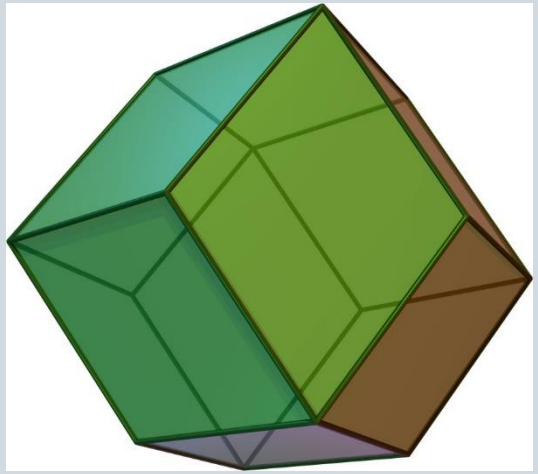
Modell II.

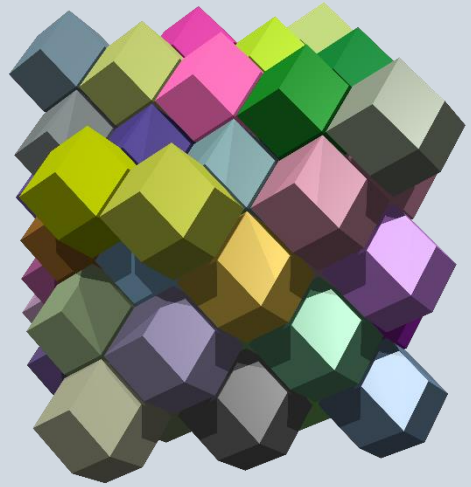
A G gráf kettős gráfjában a cellák **rombikus dodekaéderek** lesznek.

A rombikus dodekaéder egy poliéder, amelynek 12 azonos rombusz alakú lapja van.

A kettős gráf (dual graph) azt jelenti, hogy az eredeti gráf síkjai alapján új cellák jönnek létre, és ezek ebben az esetben rombikus dodekaéder formájúak.







Modell III.

Mezők és csomópontok:

A mezők, amelyeket elképzelünk, rombikus dodekaéder alakúak, és ezek passzívak, tehát nem képesek önálló mozgásra vagy számításra.

Egy **v** csomópont akkor van „burkolva” (tiled), ha egy passzív lap helyezkedik el benne. Ha nincs lap benne, akkor a csomópont üres.

Szomszédsági kapcsolatok:
A V gráf minden csomópontja 12 szomszéddal rendelkezik, hasonlóan az iránytű tizenkét irányához. Ezek a szomszédok pontosan meghatározott helyzetben helyezkednek el, és mindegyik egy-egy rombikus dodekaéderre vonatkozik.

Modell IV.

C = (T, p) konfiguráció:
Ez a halmaz tartalmazza:

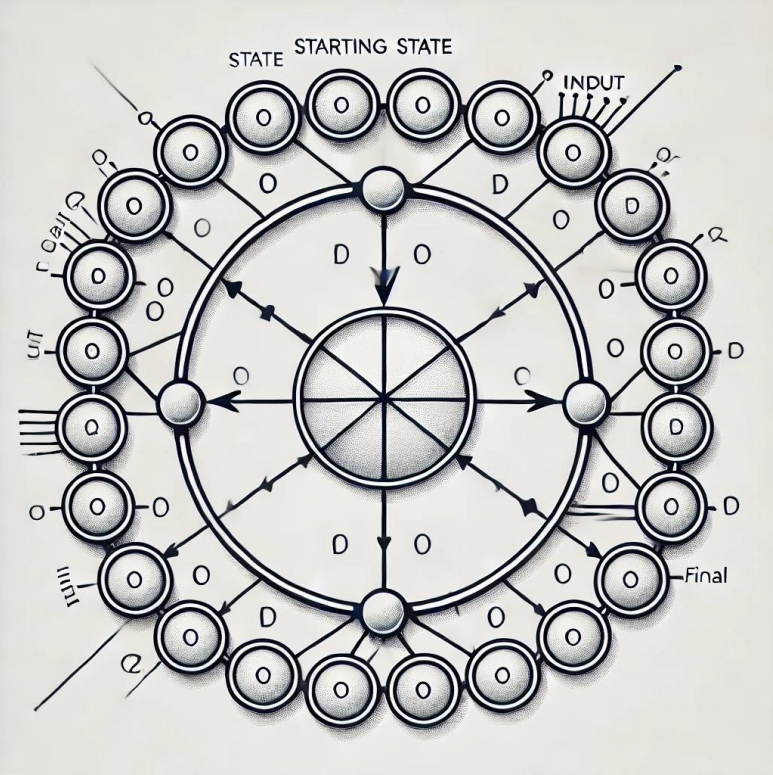
T: Az összes burkolt (tiled) csomópontot.

p: Az ügynök aktuális pozícióját.

A **C konfiguráció** összefüggő, ha:

- A gráf burkolt csomópontokból álló része, **G|T**, összefüggő, VAGY
- Az ügynök, **p**, is része a hálózatnak, és egy lapot hordoz, tehát **G|T ∪ {p}** is összefüggő.

Az ügynök, r : Nézz-Számolj-Mozdul



Probléma I.

Kezdeti konfiguráció:
Egy tetszőleges kezdetben összefüggő konfiguráció, **C0 = (T0, p0)**, ahol **p0 ∈ T**.

Az algoritmus célja egy jégcsap formáció kialakítása a konfigurációk sorozatán keresztül.

Feltételek:

Az algoritmus egy összefüggő konfigurációk sorozatát hozza létre:
C0 = (T0, p0), ..., CT = (T0, p0)

A sorozat végén, a **T** halmaz csomópontjai **jégcsap alakúak** lesznek (amit később definiálnak).

Probléma II. (Mit jelent a jégcsap?)

Szomszédos csomópontok és irányok:

Egy **v** csomópontnál:

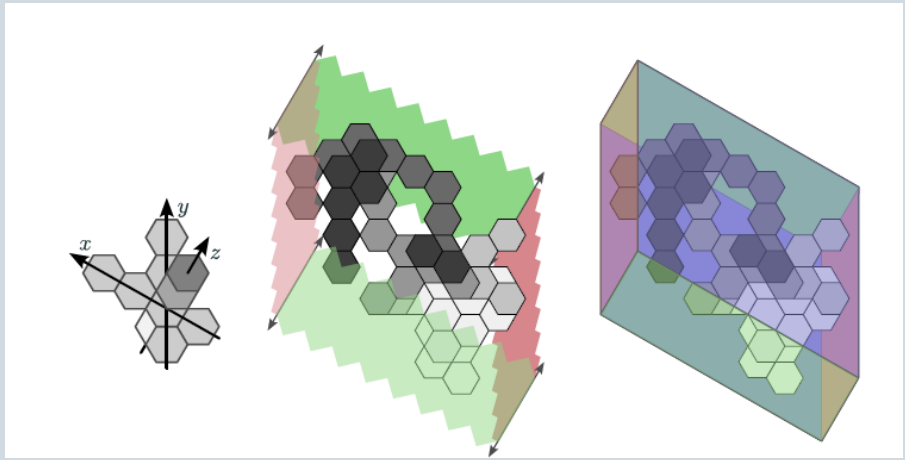
- **v + x:** az x irányban szomszédos csomópont.
- **−x:** az ellentétes irány, például **−une = dsw**.

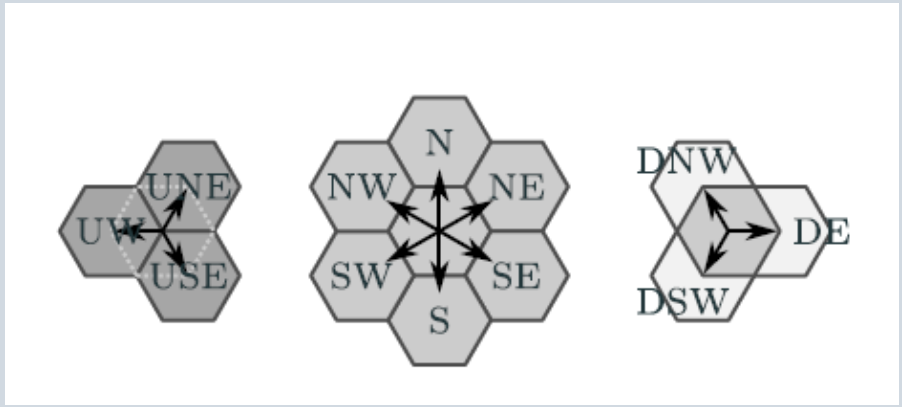
Meghatározások:

Oszlop: Az **n** és **s** irányban lévő maximális csempesor.

Sor: Az **nw** és **se** irányban elhelyezkedő csempesor.

Torony: A csempék az **une** és **dsw** irányban helyezkednek el.





Probléma III. (Mit jelent a jégcsap?)

Paralelogramma:

Egy **paralelogramma** egy maximális, egymást követő **oszlop**okból álló sorozat.

A legdélebbi csempék egy **sor**ban helyezkednek el.

Egy részben kitöltött paralelogrammában az első **oszlop** rövidebb lehet, mint a többiek.

Jégcsap definíció:

A **jégcsap** egy összefüggő **toronyhalmaz**, amelynek legfelső csempéi egy részben kitöltött paralelogrammában találhatók.

A csempék a **dsw** irányban „nőnek” le a felső paralelogrammából.

Bármely csempe, amelynek van **une** irányú szomszédja, de nincs **dsw** irányú szomszédja, eltávolítható anélkül, hogy az összefüggőség sérülne.

Ha nincs ilyen csempe, akkor a legészakibb csempe a legnyugatibb oszlopból eltávolítható.

Algoritmus

Az ügynök működése és a jégcsap formáció iteratív folyamata

Az ügynök feladata:

Az ügynök iteratívan átrendezi a **lokálisan legfelsőbb csempetöredékeket** úgy, hogy részben kitöltött **paralelogrammákat** hozzon létre.

Folyamat:

Csempék átrendezése:

- A csempék az adott rétegen belül átrendeződnek.
- Az ügynök időnként elhelyez csempéket az alsóbb rétegekben, hogy fenntartsa a **kapcsolódást**.

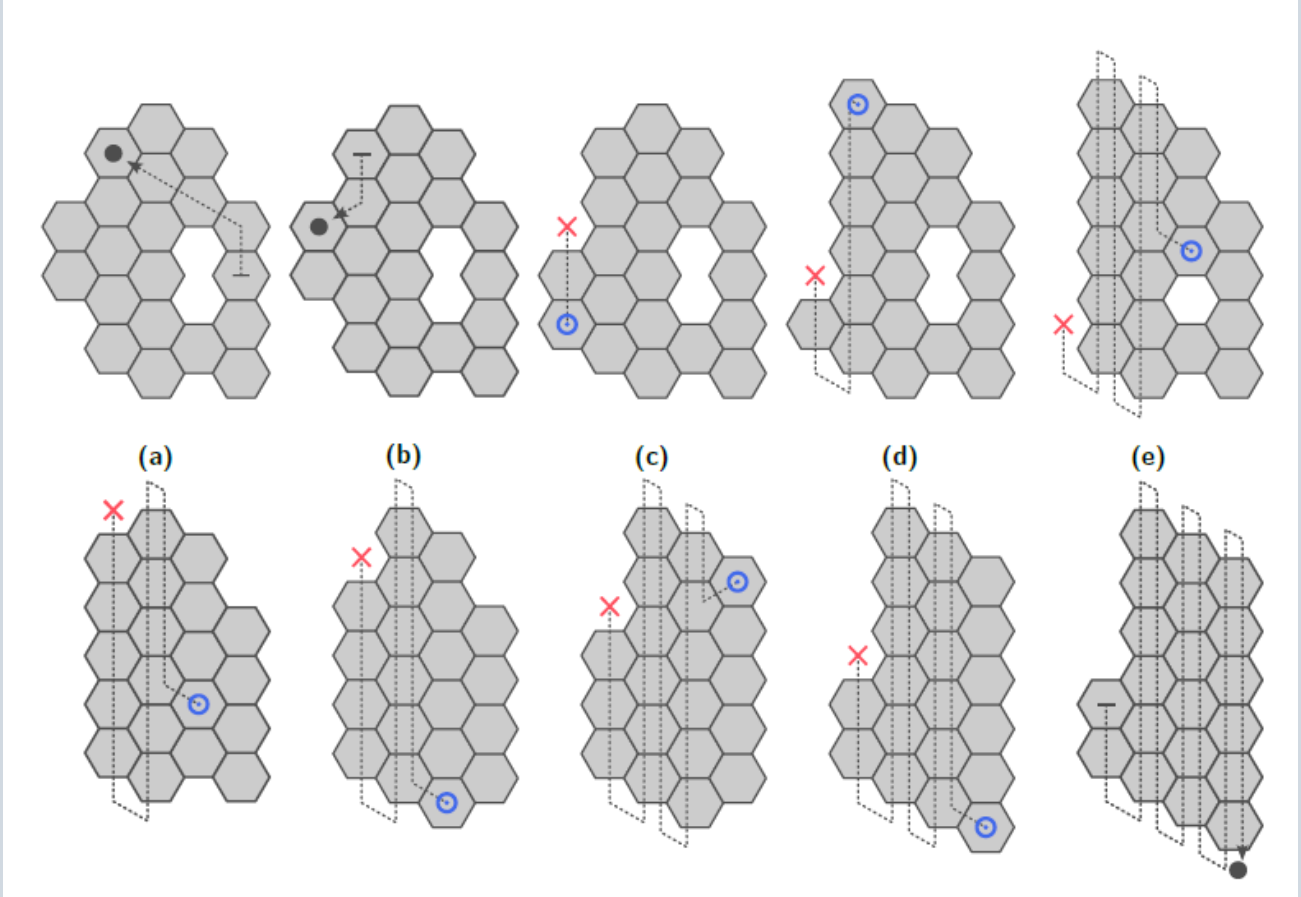
Haladás a felsőbb rétegek felé:

- Ha az ügynök felsőbb rétegekben talál csempéket, tovább halad felfelé, hogy folytassa a paralelogramma kialakítását.

Paralelogramma kialakulása után:

- A következő lépés a **projekció**, amely során minden csempe a **dsw** irányában lévő első üres csomópontra kerül.

Algoritmus szemléltetése 2D-ben



Algorithm 1 2DParallelogramFormation

```
1  while true do
2      while {p + NW, p + SW, p + N} ∩ T ≠ ∅ do move to tile at NW, SW or N
3      firstColumn ← true; run BuildPar
4      if p ∉ T then return                          ▷ terminate S of easternmost column
5      else if r carries no tile then
6          if firstColumn then
7              while p + N ∈ T do move N
8          else
9              while {p + SW, p + S} ∩ T ≠ ∅ do move to tile at SW or S
10             while {p + NW, p + SW, p + N} ∩ T ≠ ∅ do move to tile at NW, SW or N
11         pickup tile; move to tile at S, SE or NE

   procedure BuildPar
12 while p ∈ T do
13     if p + UW ∈ T or p + USE ∈ T or p + UNE ∈ T then        ▷ irrelevant in 2D
14         move to tile at UW, USE or UNE ; return
15     else if firstColumn and p + SW ∈ T then
16         move SW; return                              ▷ found more western column
17     else if p + NE ∈ T and p + SE ∉ T then
18         move SE; place tile; move NW; return         ▷ place tile below eastern column
19     else if p + N, p + SE ∈ T and p + NE ∉ T then
20         move NE; place tile; move SW; return         ▷ place tile above eastern column
21     move S
22 if p + N, p + NE, p + SE ∈ T then
23     place tile; move N                               ▷ place tile below current column
24 else if p + NE ∈ T then
25     move NE; move N; firstColumn ← false             ▷ move to top of next column
26     while p ∈ T do
27         if p + SW ∉ T and p + NW ∈ T then
28             while p + N ∈ T do move N
29             while p + NW ∉ T do move S
30             return                                   ▷ found more northern column
31         move N
32     if p + S, p + SE ∈ T then place tile             ▷ place tile above current column
33     else move S; run BuildPar
34 return
```

Algorithm 2 BUILDICICLE

```
1  while true do
2      while {p + X | X ∈ {UW, USE, UNE, NW, SW, N}} ∩ T ≠ ∅ do
3          move to tile at UW, USE, UNE, NW, SW or N
4      if G|_{N_T(p)} or G|_{N_T(p)∪{p+SE}} is connected or G|_{N_T(p)∪{p+SE+DSW}} is connected
        with p + SE + DSW ∈ T then
5          firstColumn ← true; run BUILDPAR
6          if p ∉ T then move N; run PROJECT
7          else if r carries no tile then ...        ▷ same as lines 8–15 from Algorithm 1
16     else if G|_{N_T(p)∪{p+DE}} is connected then
17         if N_T(p) = {p + DSW, p + SE} then
18             move SE + DSW; place tile; move UNE + NW; pickup tile; move SE
19         else move DE; place tile; move UW; pickup tile
20         if p + SE, p + NE ∈ T and p + S ∉ T then
21             move SE; while {p + UW, p + USE, p + SW, p + S} ∩ T = {p + S} do move S
22         else move to tile at S, SE or NE
23     else if N_T(p) = {p + DNW, p + S, p + NE} then
24         while {p + X | X ∈ {UW, USE, SW, DSW, SE, DE, S}} ∩ T = {p + S} do move S
25         if {p + X | X ∈ {UW, USE, SW}} ∩ T = ∅ then
26             if p + DE ∈ T then move N
27             move DE; place tile; move UW; pickup tile
28             while p + N ∈ T do move SE + DNW; place tile; move UW; pickup tile
29             move NE
30     else if N_T(p) = {p + DNW, p + S} then
31         while {p + X | X ∈ {UW, USE, SW, SE, DE, S}} ∩ T = {p + S} do move S
32         if {p + X | X ∈ {UW, USE, SW}} ∩ T = ∅ then
33             if {p + X | X ∈ {SE, DE}} ∩ T = ∅ then move N; run PROJECT
34             else
35                 ...                                ▷ same as lines 26–28
38             while p ∉ T do
39                 move S; if p + SE ∈ T then move SE
```

Algorithm 3

```
   procedure PROJECT
1  if p + N, p + S ∉ T then                ▷ parallelogram of height one
2      do
3          while p ∈ T do move DSW
4          place tile; while p + UNE ∈ T do move UNE
5          pickup tile
6          if p + NW ∈ T then move SW; move DNW else move DSW; return
7      while p + UNE ∈ T
8  else
9      do
10         while p + N ∈ T do move N
11         while p ∈ T do move DSW
12         place tile; while p + UNE ∈ T do move UNE
13         pickup tile
14         if p + S ∈ T then move S
15         else if p + NW ∈ T then move NW
16         else move DSW; return
17     while p ∈ T
```

Algoritmus lemmák

Futási idő: $O(n^3)$ lépések

Garancia:

Az algoritmus futási ideje garantáltan **$O(n^3)$** lépés, ahol **n** a csempék száma.

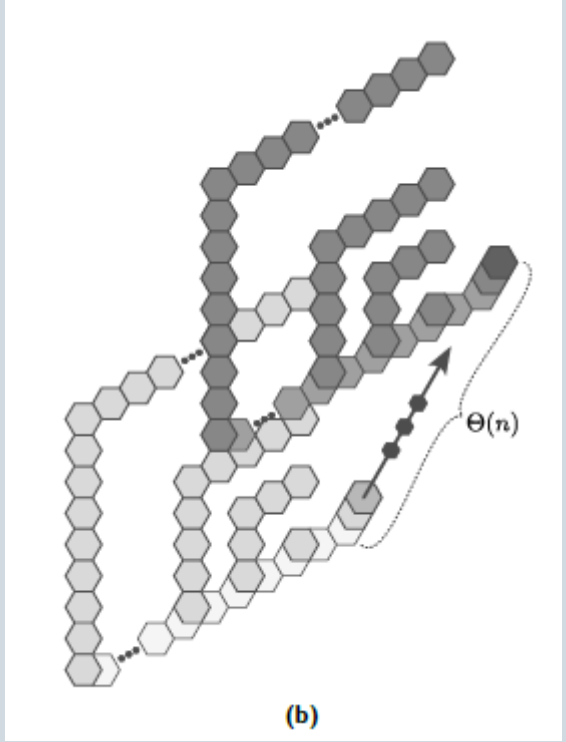
A teljesítmény megegyezik a 3D **vonalformációs** algoritmuséval, amely egy hasonló korábbi megközelítés volt.

Részletezés:

Az algoritmus garantálja, hogy bármely kezdeti konfigurációból kiindulva képes egy **jégcsap formát** elérni.

A csempék mozgatásával és rendezésével a jégcsap alakzat kialakítása gyorsabb és hatékonyabb, mint a **vonal formájú** alakzat.

A jégcsap formáció előnye: csökkenti az alakzat átmérőjét, és több eltávolítható csempét tartalmaz, ezáltal gyorsítva a folyamatot.



Konvergencia a jégcsap struktúrához

Garancia:

Az algoritmus biztosítja, hogy bármelyik összefüggő kezdeti csempeszerkezet **konvergál egy jégcsap alakhoz**, függetlenül a kezdeti állapottól.

Részletezés:

Az algoritmus minden lépése során fenntartja a csempék **kapcsolódását**, így biztosítva a folyamatos összefüggőséget.

Az ágens folyamatosan átalakítja a csempék elrendezését, amíg végül egy **jól meghatározott jégcsap struktúrát** képez.

A jégcsap forma előnyei:

- Az ágens könnyebben navigálhat a csempék között.
- Biztonságosan azonosíthatja az **eltávolítható csempéket**, anélkül, hogy megszakítaná a szerkezet összefüggőségét.

Konnektivitás fenntartása

Garancia:

Az algoritmus alapvető követelménye, hogy **biztosítja a csempék összefüggőségét** az egész folyamat során.

Minden csempe áthelyezése és mozgatása közben az algoritmus folyamatosan ellenőrzi, hogy az átalakított szerkezet **összefüggő maradjon**.

Ez különösen fontos olyan környezetekben, mint a folyadékban vagy alacsony gravitációs környezetben, ahol a csempék közötti kapcsolat megszakadása nemkívánatos következményekkel járhat.

Részletezés:

Az algoritmus garantálja, hogy bármely eltávolított vagy mozgatott csempe után a szerkezet **összefüggő marad**.

Minden csempe mozgatása előtt az algoritmus ellenőrzi, hogy van-e **biztonságosan eltávolítható** csempe, amelynek eltávolítása után a többi csempe kapcsolódása nem szakad meg.

A jégcsap forma előnyei és eltávolítható csempék

Garancia:

A jégcsap struktúra több **eltávolítható csempét** tartalmaz, így az algoritmus könnyebben talál olyan csempéket, amelyeket mozgatni lehet anélkül, hogy megsértené a szerkezet integritását.

Részletezés:

Az eltávolítható csempék kulcsfontosságúak az algoritmus számára, mert lehetővé teszik az ügynök számára, hogy átrendezze a csempéket és kialakítsa a kívánt struktúrát a szerkezet stabilitásának megzavarása nélkül.

A **jégcsap formáció** több eltávolítható csempét kínál, mint a vonal formáció, így gyorsabb és hatékonyabb átrendezést biztosít.

A vonal formáció kevesebb eltávolítható csempével rendelkezik, ami:

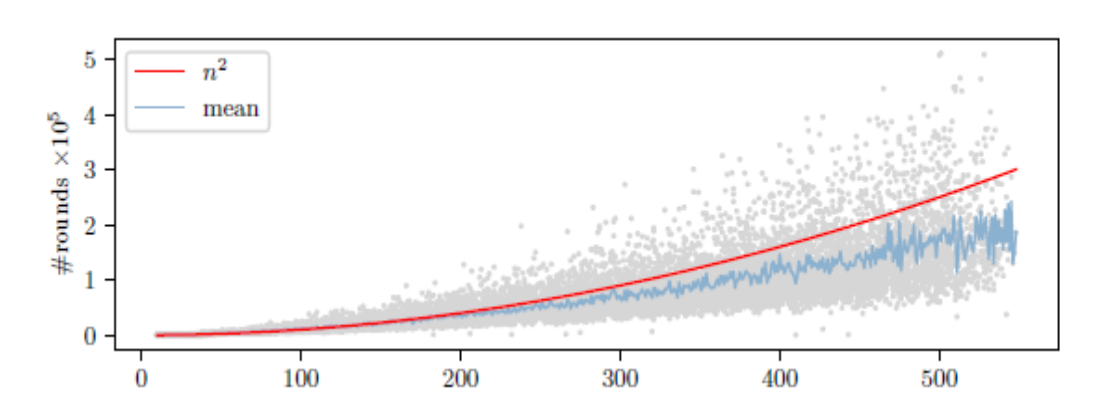
- Hosszabb keresési időt eredményez.
- Nagyobb átmérőt von maga után.

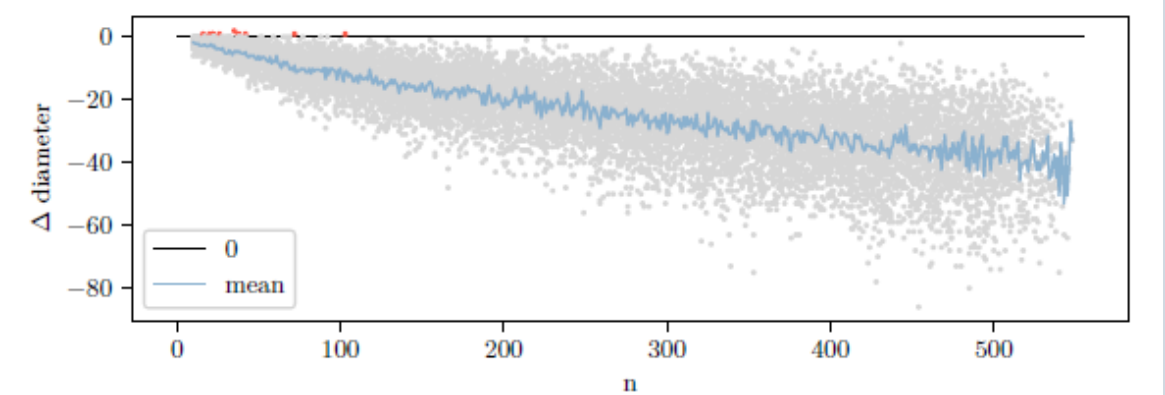
A **jégcsap forma** hatékonyabb, mivel optimalizálja az eltávolítható csempéket és csökkenti a keresési időt.

Futási idő optimalizálása és szimulációs eredmények

Bár a legrosszabb esetben az algoritmus futási ideje **$O(n^3)$** lépés, a szimulációk alapján az algoritmus futási ideje gyakran **$O(n^2)$** körül mozog.

Valós körülmények között az algoritmus gyorsabban működik, mint amit a legrosszabb eset analízise alapján várnánk.





Köszönöm a figyelmet!