

Typed DLA

Chris Mueller

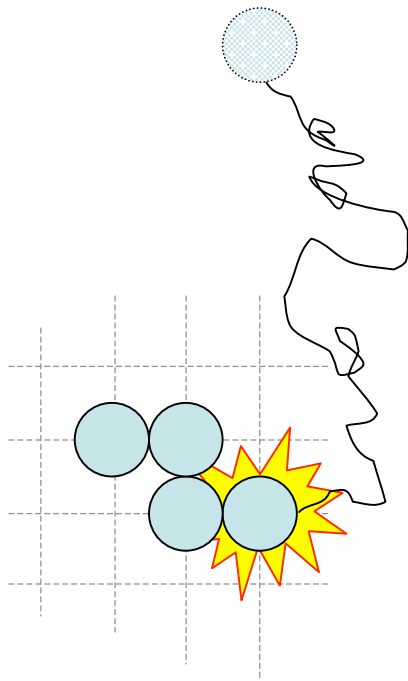
P700: Fractals and Pattern Formation

Project Study

March 5, 2004

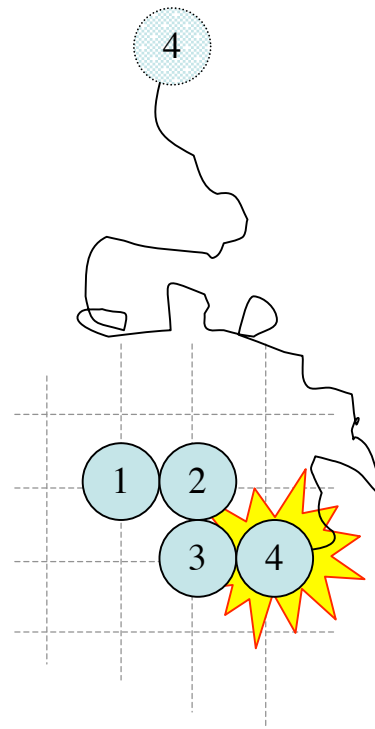
Traditional DLA

A walker can stick to any point in the structure.

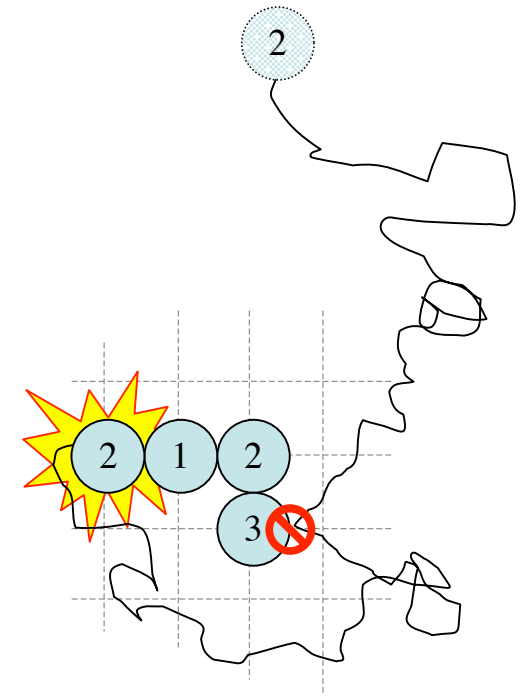


Typed DLA

Each walker and point has a type. Random walkers can only stick to points in the structure with compatible types.



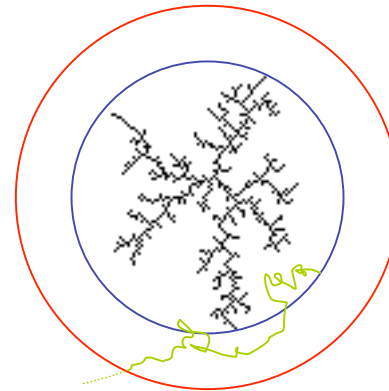
A walker of type 4 can stick to a point of type 3.



A walker of type 2 cannot stick to a point of type 3! It keeps walking until it hits a point of type 1 and sticks.

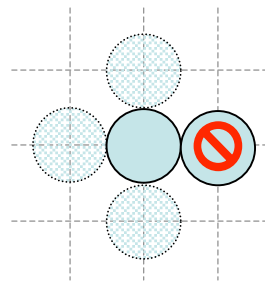
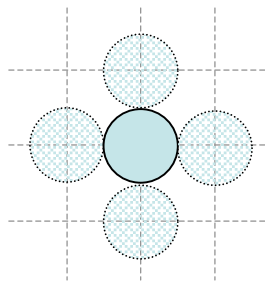
Typed DLA Algorithm

```
walker = ReleaseWalker();
while not done:
    Step(walker)
    if WanderedOff(walker):
        done = true
    else if Hit(lattice, walker):
        if Stick(lattice, walker):
            Add(lattice, walker)
            done = 1
```

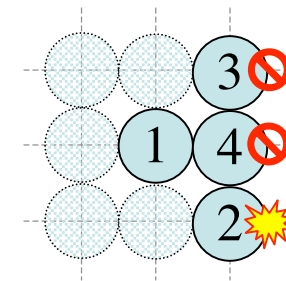
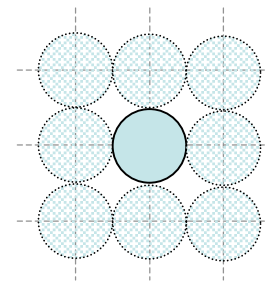


`ReleaseWalker()` creates a walker at a random point on a circle at the edge of the current structure.

`WanderedOff()` determines if the walker exited the far boundary.



`Step()` moves the walker to a random, unoccupied location one lattice point up, down, left, or right.



`Hit()` looks at all points one pixel away, including the diagonals. If a hit is detected, `Stick()` tests to see if the walker can stick to any of the adjacent points. `Add()` then connects the point to the structure.

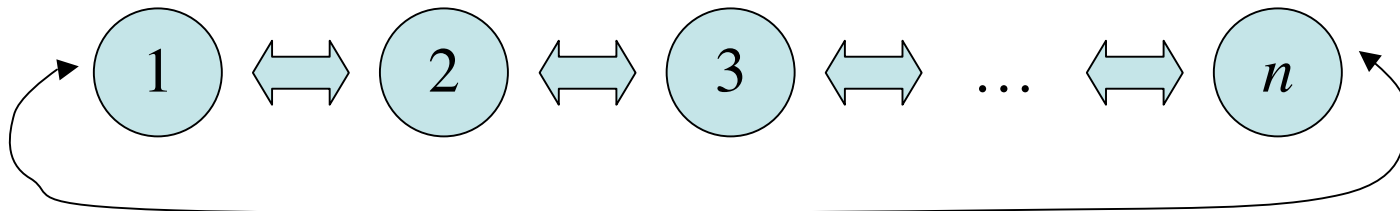
Sticking Rules

The type of each walker and point is a positive integer. Any function $F:N \times N \rightarrow B$ where B is the set $\{True, False\}$ is valid as a sticking rule.

The rule used for this study is:

$$\text{Stick}(\text{type}(\text{walker}), \text{type}(\text{point})) = \begin{cases} True & t_walker = (t_point - 1) \% n \text{ or} \\ & (t_point + 1) \% n \\ False & \text{Otherwise} \end{cases}$$

where n is the number of types. Basically, types that are adjacent numerically can stick to each other, with 1 and n also sticking.



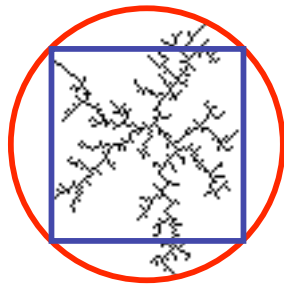
Experiments!

Questions:

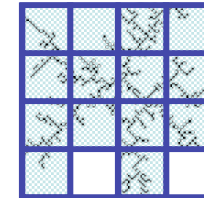
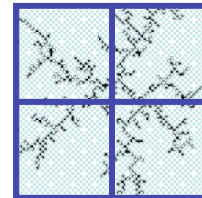
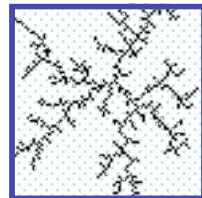
- Does varying the number of types change the behavior of the DLA?
- How does the fractal dimension vary with the number of types?

Method:

- Simulate the typed DLA for different amounts of types on different sized $n \times n$ grids.
- Compute the fractal dimensions for each grid size and number of types using the Box Counting algorithm:



Sub-sample the DLA to avoid blank areas at the edge



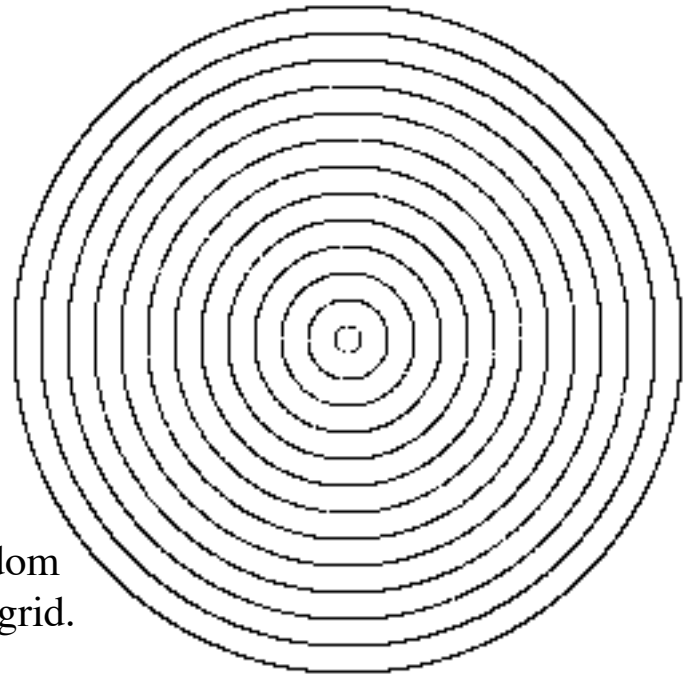
Starting with a box the size of the sample and dividing each box into 4 new boxes at each step, count the number of boxes that the structure intersects. Compute $D = \text{avg}(\log_2(\text{count}/\text{count}_{i-1}))/\log_2(\text{size}_0)$.

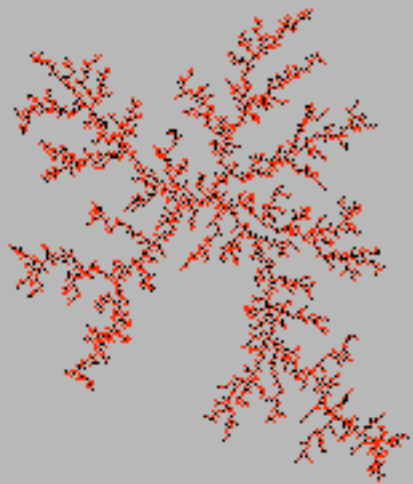
Setup

- Grid Sizes: 32, 64, 128, 256, 512, 1024
- Number of types: 1, 3, 5, 10, 20
- Number of samples per grid size/type: 30
- Total number of experiments: 900
- Development Methodology
 - Rapid prototype in Python with real time visualization
 - Experiments implemented in C++, Python used for visualizations
- Hardware
 - 32 - 512 grid experiments
 - Dual 2.0 GHz G5 workstation
 - 1024 grid experiments
 - AVIDD-B

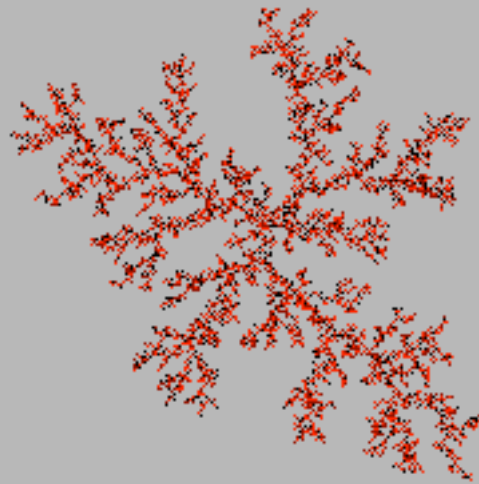
Can you guess what happened?

Release points for random walkers on a 256x256 grid.

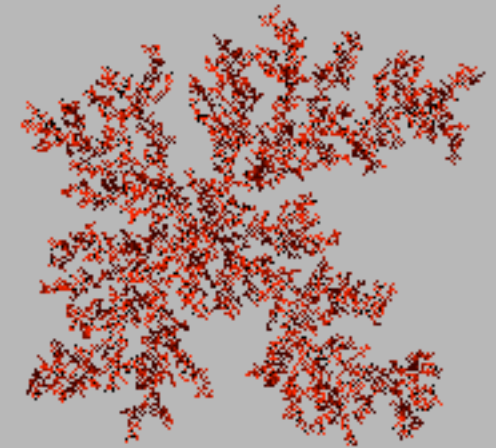




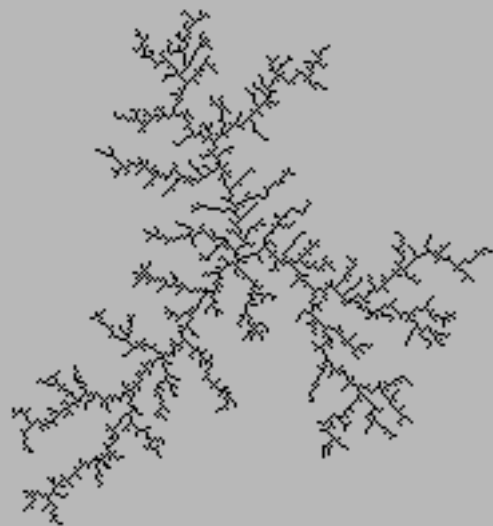
3 Types, $D = 1.56$, 14285 Walkers



5 Types, $D = 1.61$, 13822 Walkers



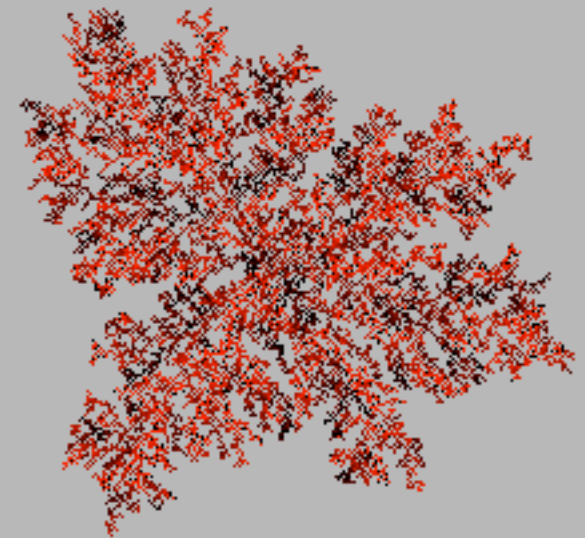
10 Types, $D = 1.67$, 27160 Walkers



1 Type, $D = 1.53$, 10992 Walkers

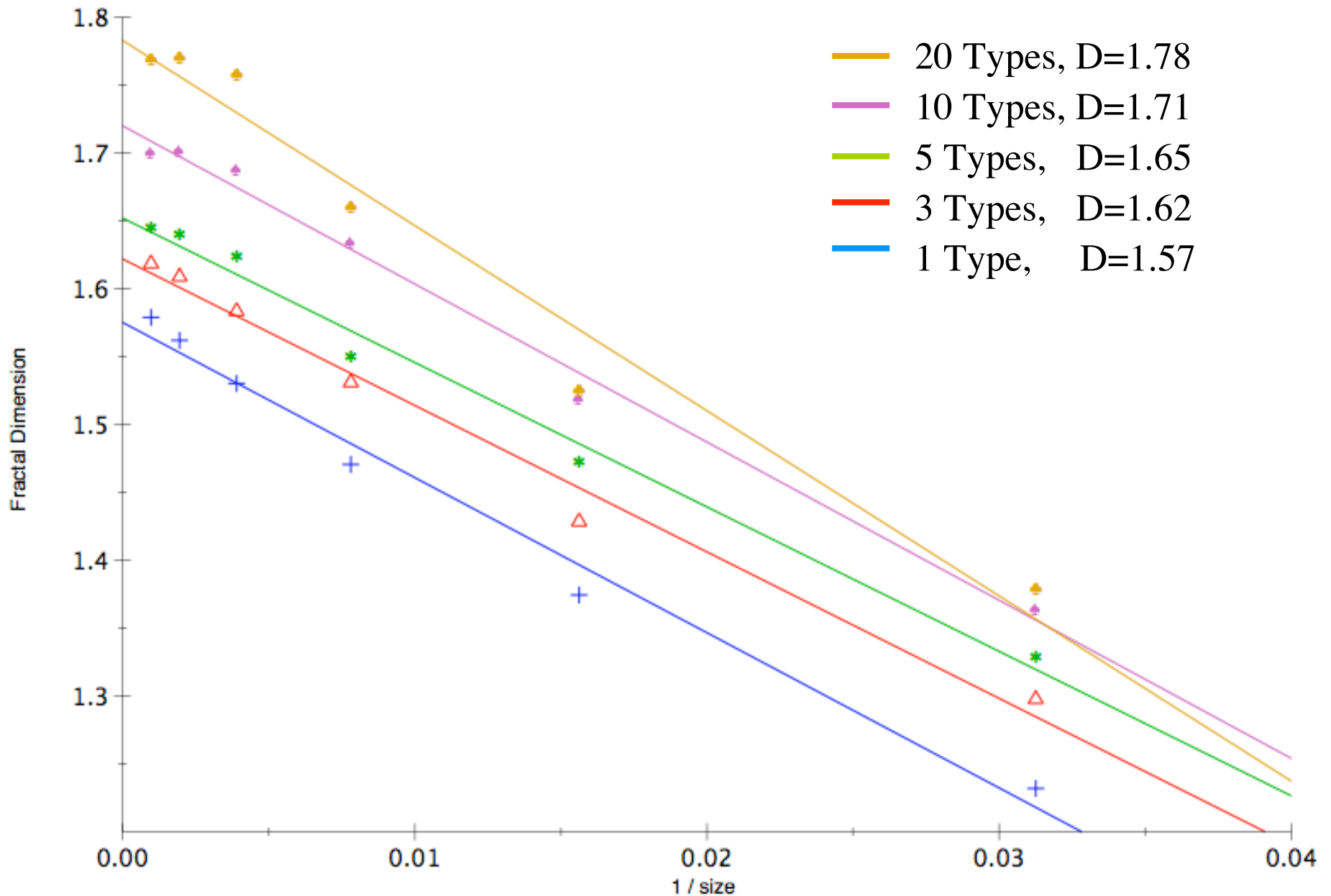
Typed DLAs (256x256)

The color of each pixel
corresponds to its type.

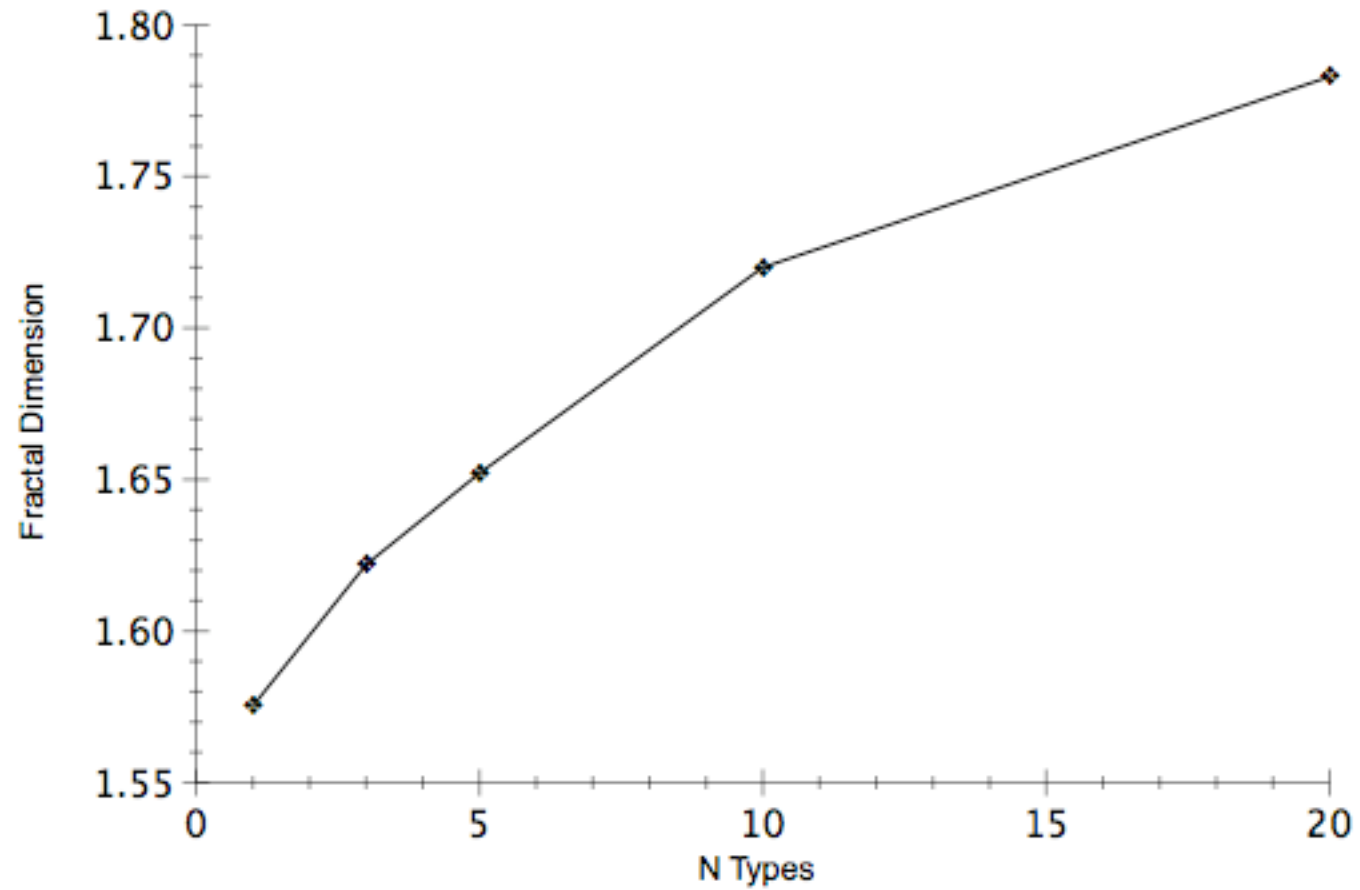


20 Types, $D = 1.74$, 42966 Walkers

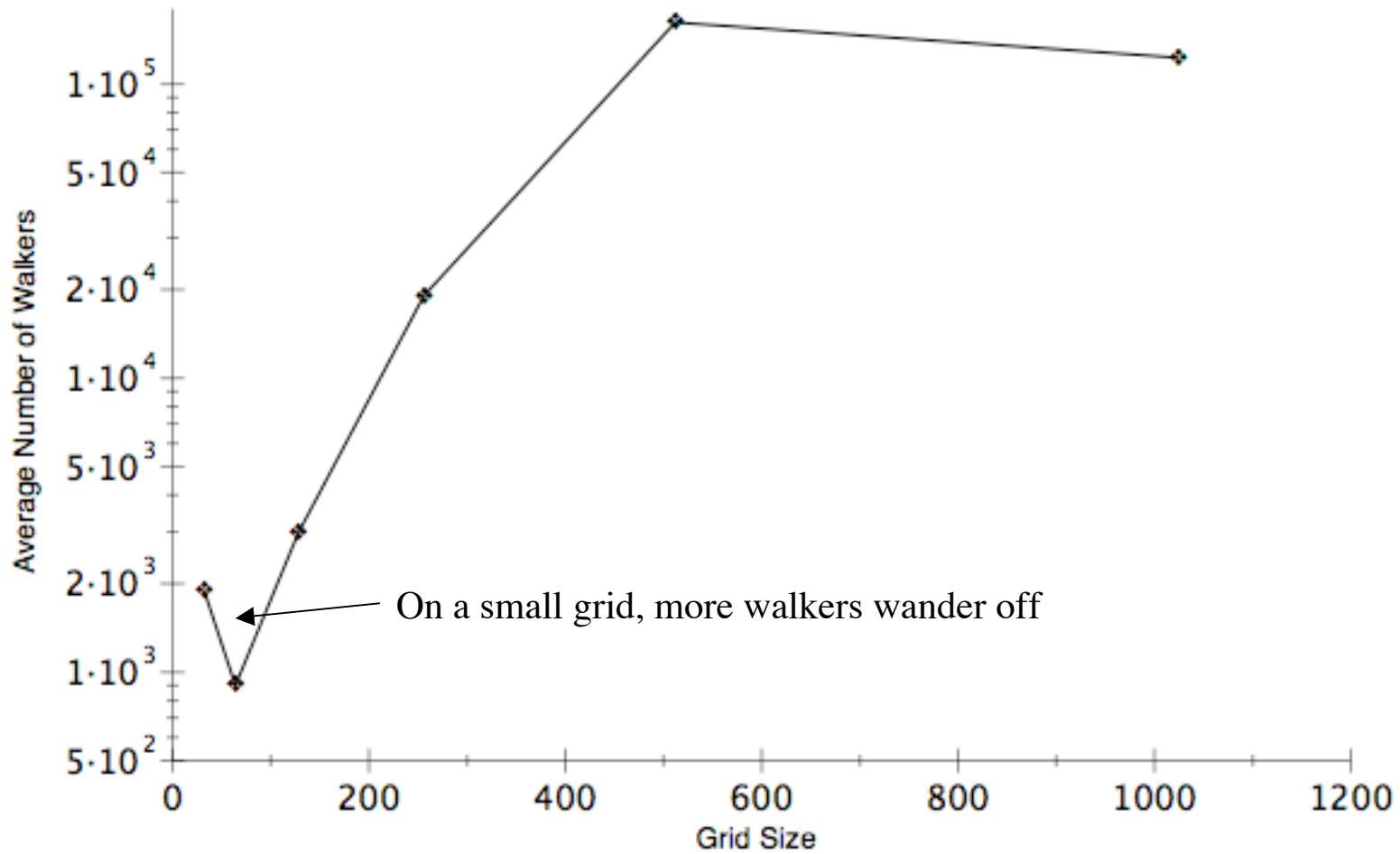
Fractal Dimensions



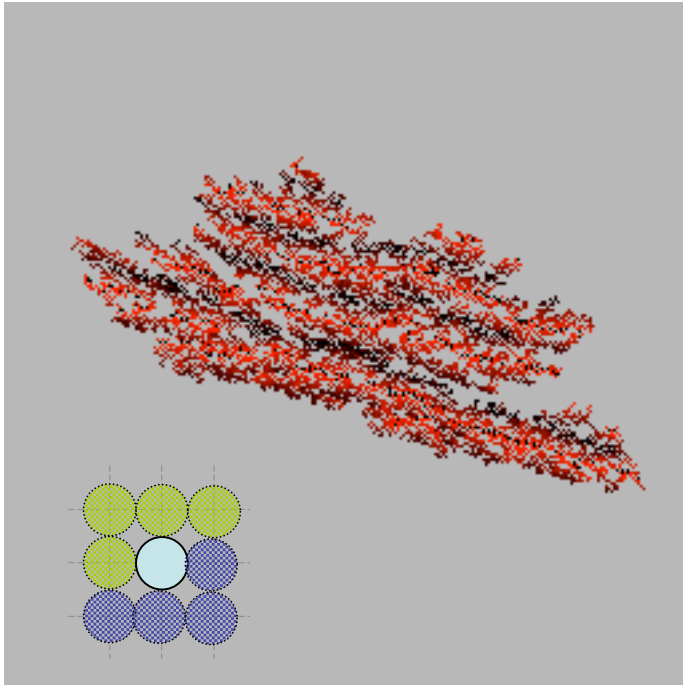
Fractal Dimensions (2)



Average Number of Walkers

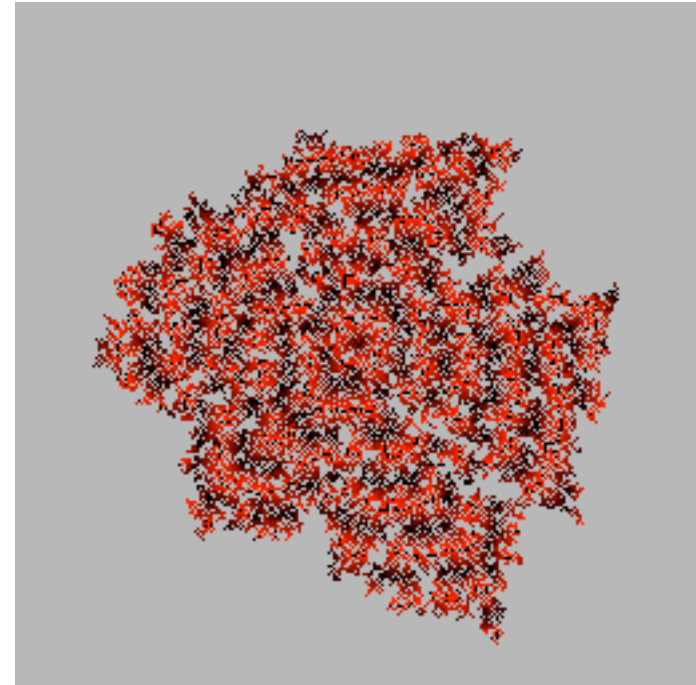


Fun Stuff!



(a)

$$(a) \text{ Stick}(\text{walker}, \text{point}) = \begin{cases} \textit{True} & \text{type}(\text{walker}) = ((\text{type}(\text{top}(\text{point})) - 1) \% n \text{ or} \\ & (\text{type}(\text{bottom}(\text{point})) + 1) \% n) \\ \textit{False} & \text{Otherwise} \end{cases}$$



(b)

$$(b) \text{ Stick}(\text{walker}, \text{point}) = \begin{cases} \textit{True} & \text{type}(\text{walker}) = ((\text{type}(\text{point}) + 1) \% n) \\ \textit{False} & \text{Otherwise} \end{cases}$$