



Parallel & Distributed: DIFFUSION LIMITED AGGREGATION (MPI)

Nguyễn Tổng Minh - 20204885

Nguyễn Xuân Thái Hòa - 20204878

Hoàng Xuân Việt - 20200662

Lý Nhật Nam - 20204886

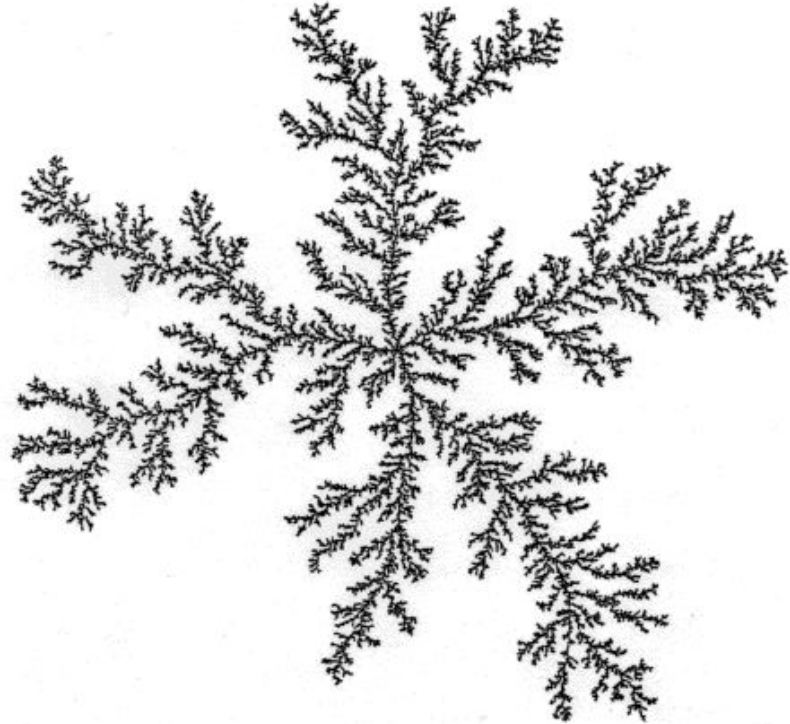
Nguyễn Công Đạt - 20200137

TEAM 2 present
SOICT (2022)



INTRODUCE THE PROBLEM

TEAM 2 present
SOICT (2022)



Time-independent Diffusion



In many situations, we may solely interested in the steady state, not much into the transient behaviours:

$$\nabla^2 c = 0 .$$

Reduce the problem into solving time-independent Laplace equation.

The concentration is no longer depends on the time variable, so by substituting the finite difference for the spatial derivatives into the **Laplace equation**:

$$c_{l,m} = \frac{1}{4} [c_{l+1,m} + c_{l-1,m} + c_{l,m+1} + c_{l,m-1}] , \forall (l,m) .$$

(Time-independent Diffusion Equation)

Limited Diffusion Aggregation (1)

"Diffusion" because the particles forming the structure wander around randomly before attaching themselves ("Aggregating") to the structure. "Diffusion-limited" because the particles are in low concentrations so they don't come in contact with each other and *the structure grows one particle at a time rather than by chunks of particles.*

Algorithm 1 DLA algorithm

time-independent diffusion eq.: $\Delta c = 0$

loop till convergence:

1. Solve $\Delta c = 0$. Object be sink ($c = 0$).
 2. Let object grow.
-

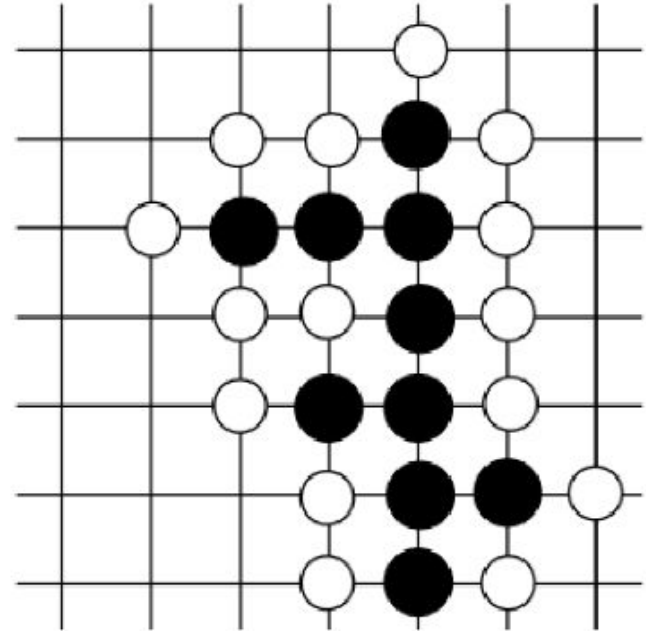


Figure 1: The object and possible growth sites.

Limited Diffusion Aggregation (2)

1. Determine growth candidates.
2. Determine growth probabilities.
3. Grow candidates.

$$p_g((i, j) \in \circ \rightarrow (i, j) \in \bullet) = \frac{(c_{i,j})^\eta}{\sum_{(i,j) \in \circ} (c_{i,j})^\eta}.$$

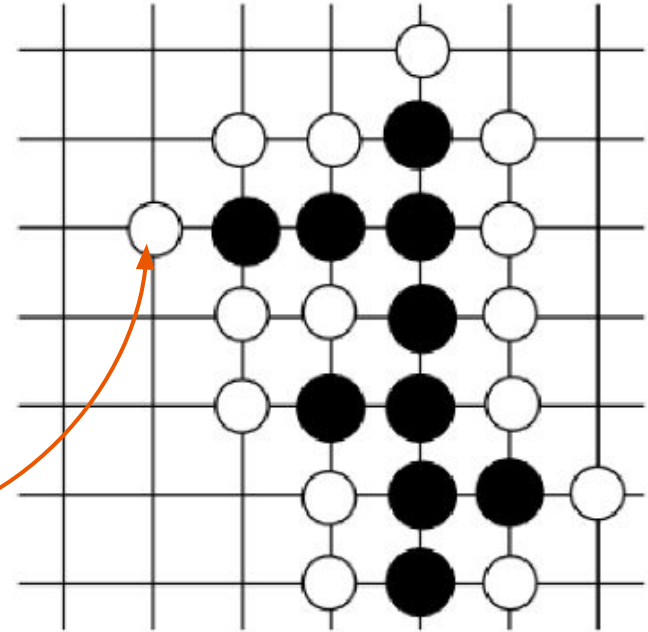
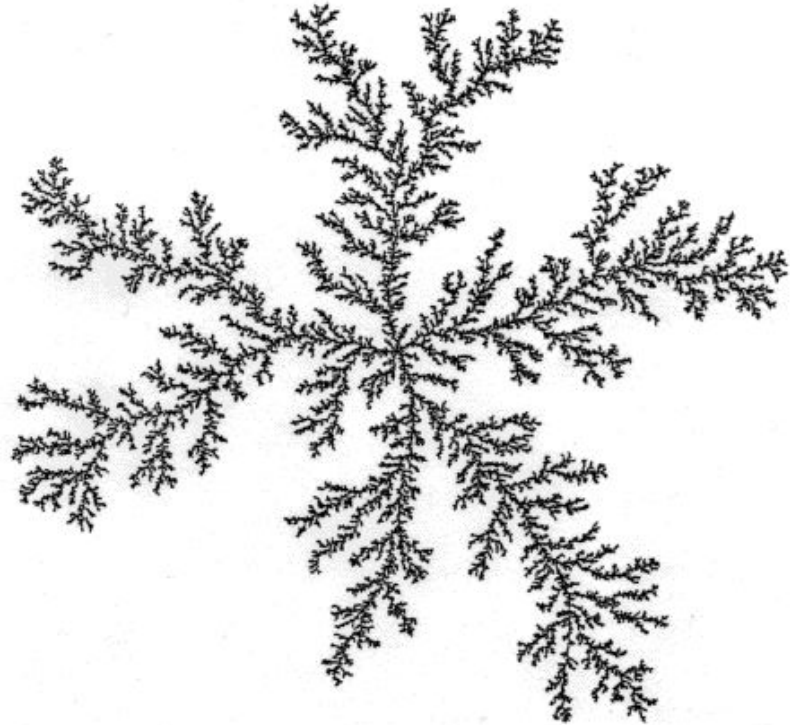


Figure 1: The object and possible growth sites.

ALGORITHMS WITH PARALLEL DESIGN

TEAM 2 present
SOICT (2022)



Successive Over Relaxation

```
/* Jacobi update, square domain, periodic in x, fixed */
/* upper and lower boundaries */
do {
     $\delta = 0$ 
    for i=0 to max {
        for j=0 to max {
            if( $c_{ij}$  is a source)  $c_{ij}^{(n+1)} = 1.0$ 
            else if( $c_{ij}$  is a sink)  $c_{ij}^{(n+1)} = 0.0$ 
            else {
                /* periodic boundaries */
                west = (i==0) ?  $c_{max-1,j}^{(n)}$  :  $c_{i-1,j}^{(n)}$ 
                east = (i==max) ?  $c_{1,j}^{(n)}$  :  $c_{i+1,j}^{(n)}$ 
                /* fixed boundaries */
                south = (j==0) ?  $c_0$  :  $c_{i,j-1}^{(n)}$ 
                north = (j==max) ?  $c_L$  :  $c_{i,j+1}^{(n)}$ 
                 $c_{ij}^{(n+1)} = 0.25 * (west + east + south + north)$ 
            }
        }
        /* stopping criterion */
        if( $|c_{ij}^{(n+1)} - c_{ij}^{(n)}| > tolerance$ )  $\delta = |c_{ij}^{(n+1)} - c_{ij}^{(n)}|$ 
    }
}
while ( $\delta > tolerance$ )
```

SOR can be thought of as a smoothed version of Gauss-Seidel iterative method by using momentum.

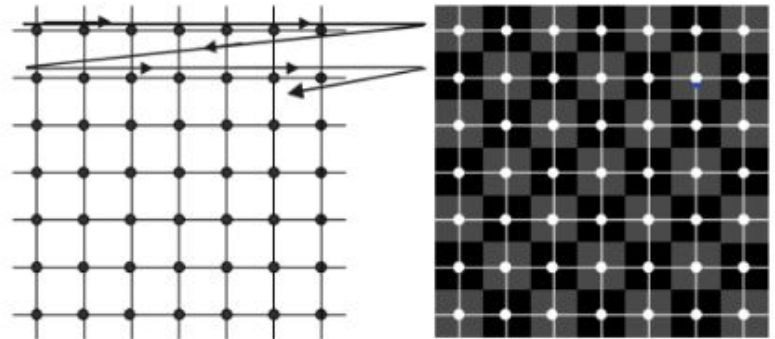
$$c_{l,m}^{(n+1)} = \frac{\omega}{4} [c_{l+1,m}^{(n)} + c_{l-1,m}^{(n+1)} + c_{l,m+1}^{(n)} + c_{l,m-1}^{(n+1)}] + (1-\omega) c_{l,m}^{(n)}$$

Parallel Design

- ❖ Each processor handles some row of the grid, with red-black ordering.

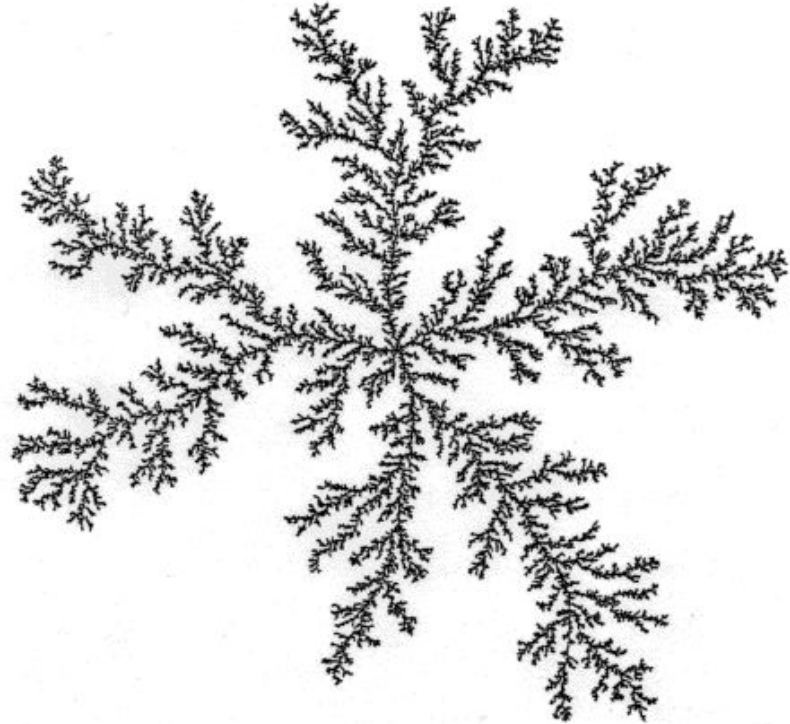
```
/* only the inner loop of the parallel Gauss-Seidel method with */  
/* Red Black ordering */  
do {  
    exchange boundary strips with neighboring processors;  
    for all red grid points in this processor {  
        update according to Gauss-Seidel iteration;  
    }  
    exchange boundary strips with neighboring processors;  
    for all black grid points in this processor {  
        update according to Gauss-Seidel iteration;  
    }  
    obtain the global maximum  $\delta$  of all local  $\delta_i$  values  
}  
while ( $\delta > \text{tolerance}$ )
```

Algorithm 4: The pseudo code for parallel Gauss-Seidel iteration with red-black ordering.

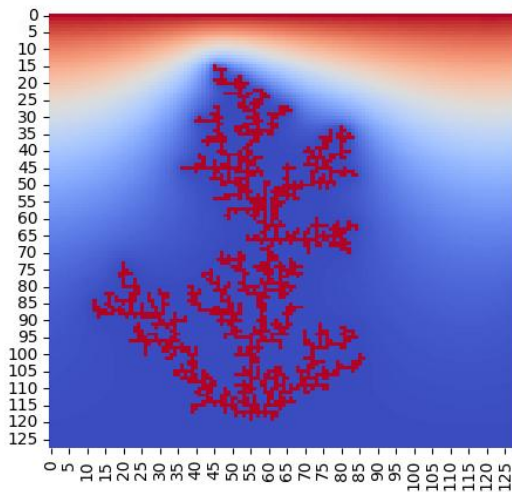


EXPERIMENTS AND RESULTS

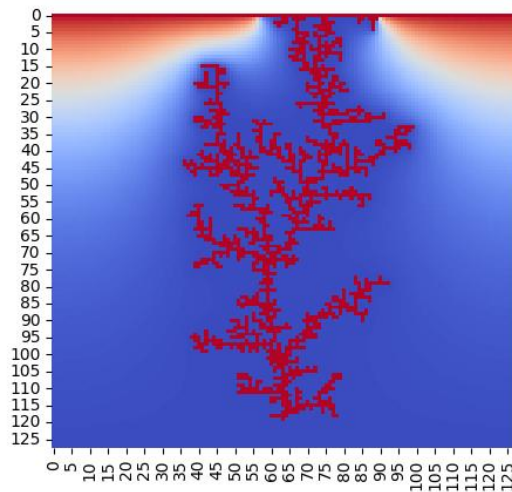
TEAM 2 present
SOICT (2022)



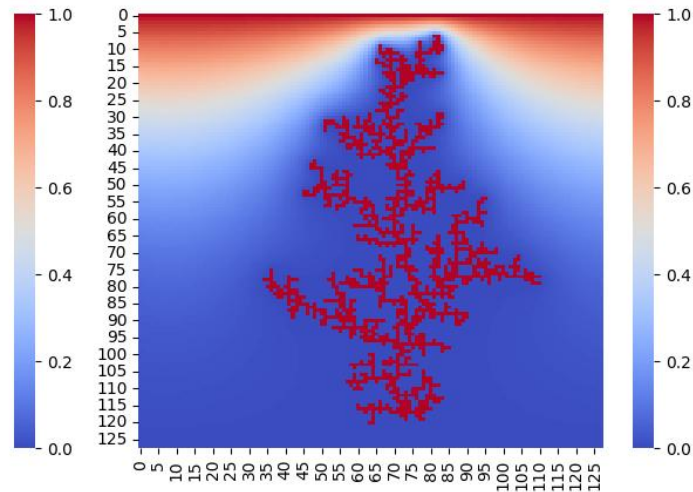
Object Initialization



(mid-right)

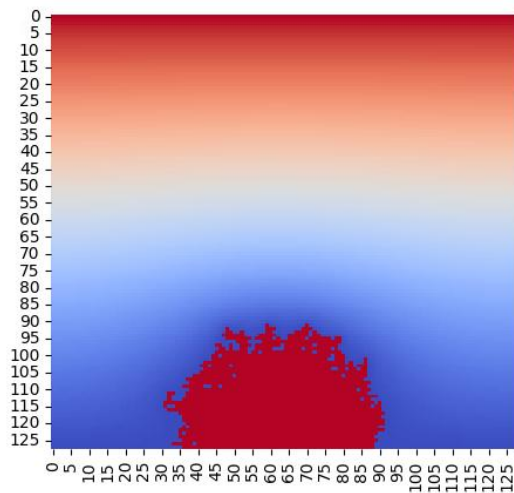


(mid-middle)

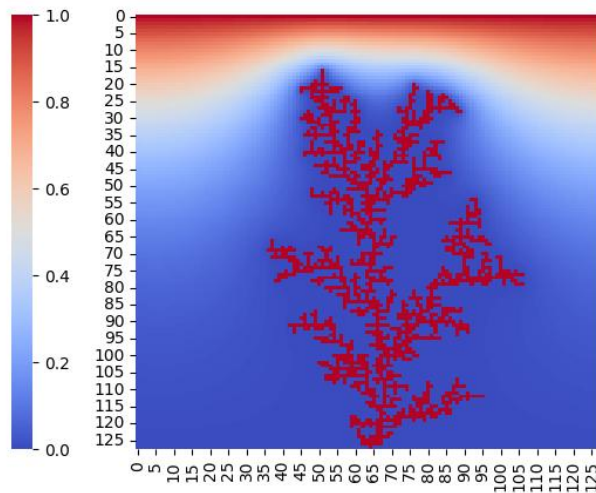


(mid-left)

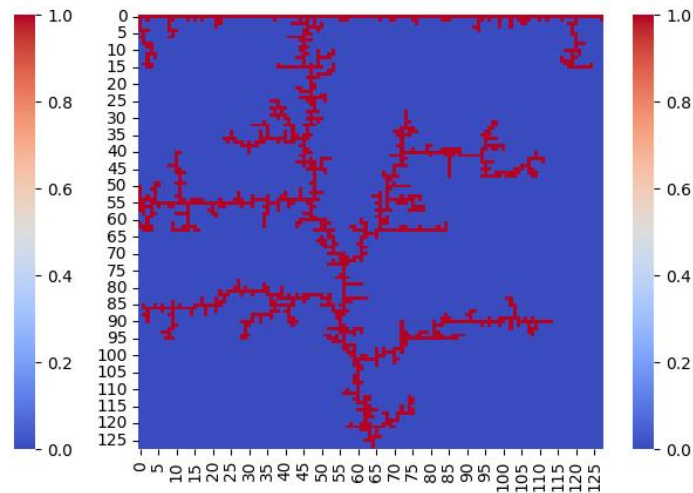
Theta Initialization



($\theta=0$)

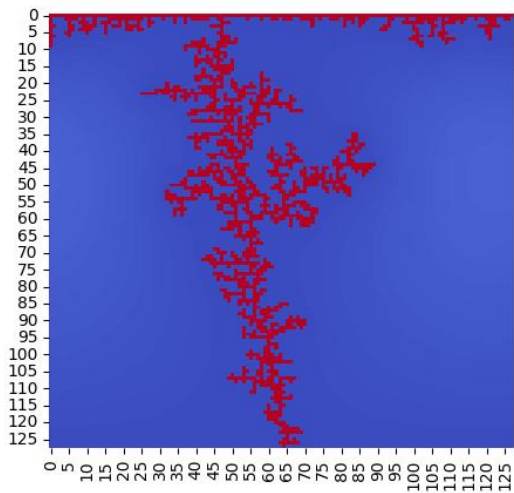


($\theta=1$)

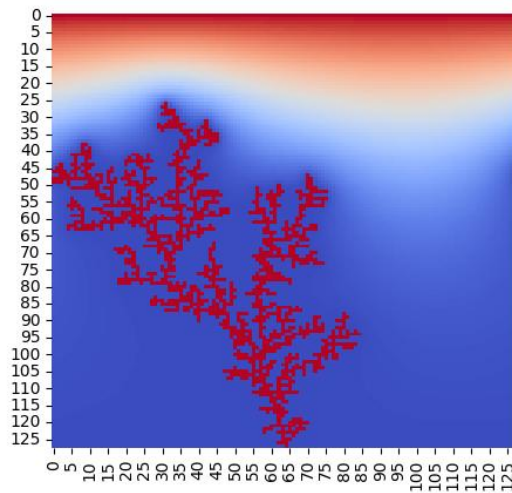


($\theta=2$)

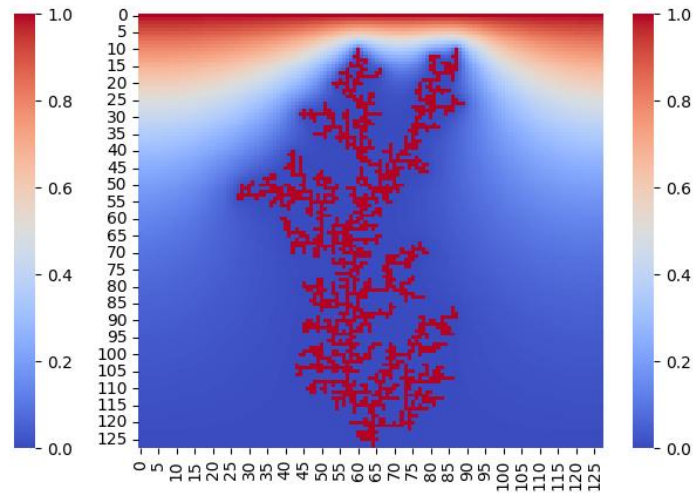
Omega Initialization



($\omega = 1.3$)

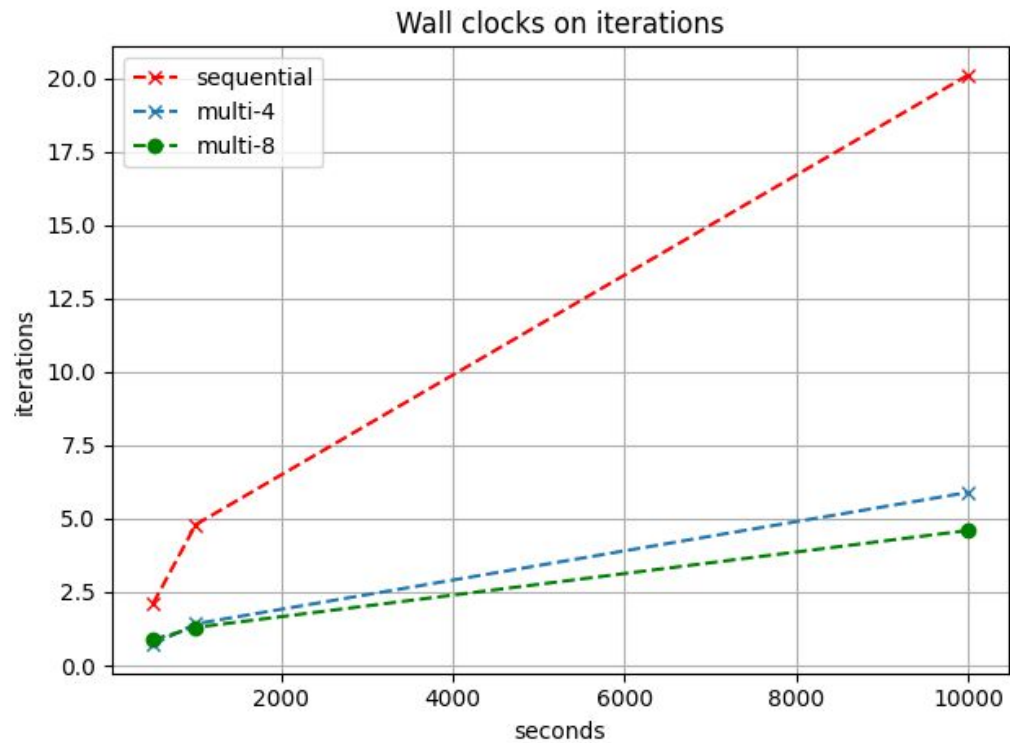


($\omega = 1.6$)



($\omega = 1.9$)

Wall Clock





Parallel & Distributed: DIFFUSION LIMITED AGGREGATION

Nguyễn Tổng Minh - 20204885

END