Diffusion Limited Aggregation

Group 7

Pham Quang Hieu | 20194432

Nguyen Van Thanh Tung | 20190090

Nguyen Vu Thien Trang | 20194459

Table of contents

Introduction

- ❖ In many cases in practice of diffusion problem we only care about final steady state of concentration.
- ❖ Thus many problem reduced to solving time independent Laplace equation.

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

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

Problem formulation

- ❖ Diffusion Limited Aggregation (DLA) is a model for non-equilibrium growth, where growth is determined by diffusion particles.
- ❖ Problem solving steps:
  - ❖ Solve Laplace equation to get distribution of nutrients, assume that the object is a sink (i.e $c = 0$ on the object)
  - ❖ Let the object grow
  - ❖ Go back to the first step.

Problem formulation

- ❖ Algorithm for growing:
  - ❖ Determine grow candidates
  - ❖ Determine growth probabilities
  - ❖ grow

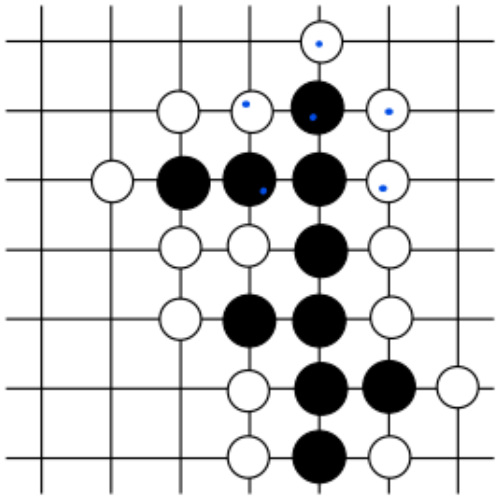


Figure 18: The object and possible growth sites.

The probability for growth at each of the growth candidates is calculated by

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

Problem formulation

- ❖ We choose to apply Successive Order Relaxation (SOR) method with MPI.
- ❖ SOR can be thought of as a smoothed version of Gauss Seidel Iterative method by using momentum.

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

Problem formulation

- ❖ Some boundary conditions in case of 2D grid:
  - ❖ Periodic in columns direction, with a period equals to N (width of the grid).
  - ❖ Constant in row direction.

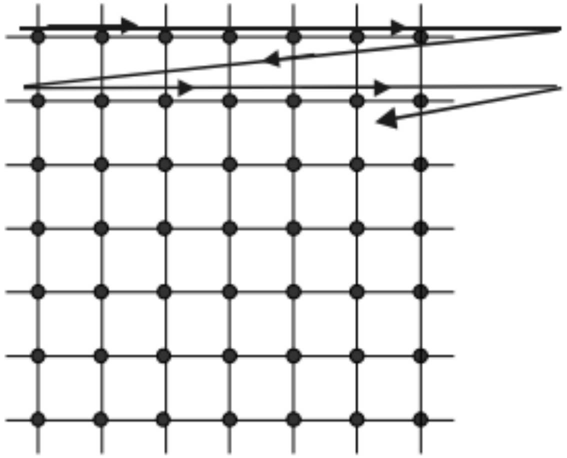
Implementation details

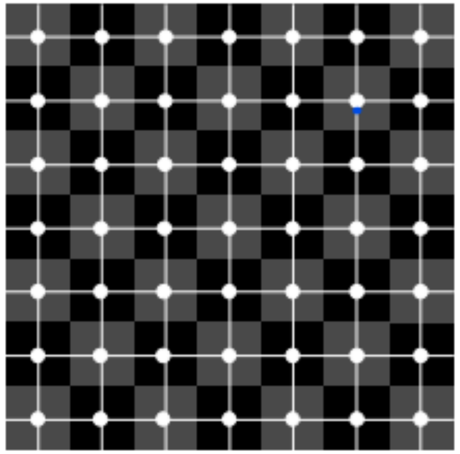
Implementation details

❖ 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 δ of all local δ_l values
   }
   while (δ > tolerance)
```

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





Demo

Thank you