Nonlinear Optimization to Generating Non-overlapping Random Dot Patterns

Takashi Imamichi, HIdetoshi Numata,
HIdeyuki Mizuta, Tsuyoshi Idé
IBM Research – Tokyo

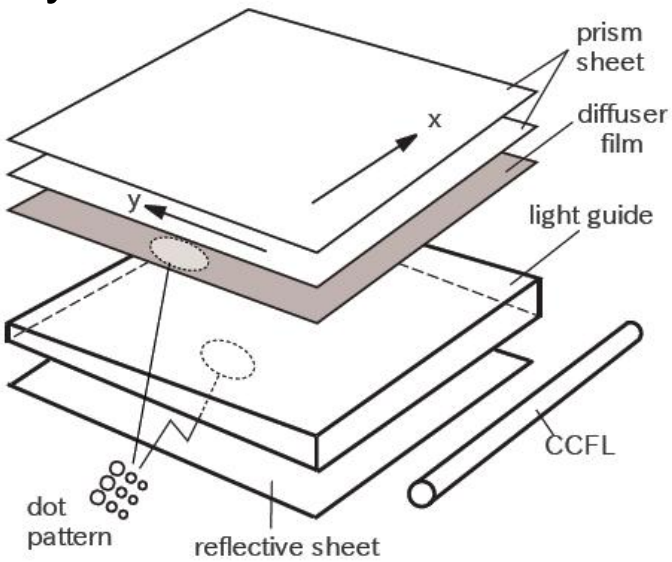
Winter Simulation Conference 2011
Phoenix, Arizona
11-14 December 2011

Outline

- Motivation
  - Requirements of dot patterns
- Related Work
  - Molecular-dynamics based methods
- Optimization-based Algorithm
  - Unconstrained nonlinear programming problem
- Experiments
- Conclusion

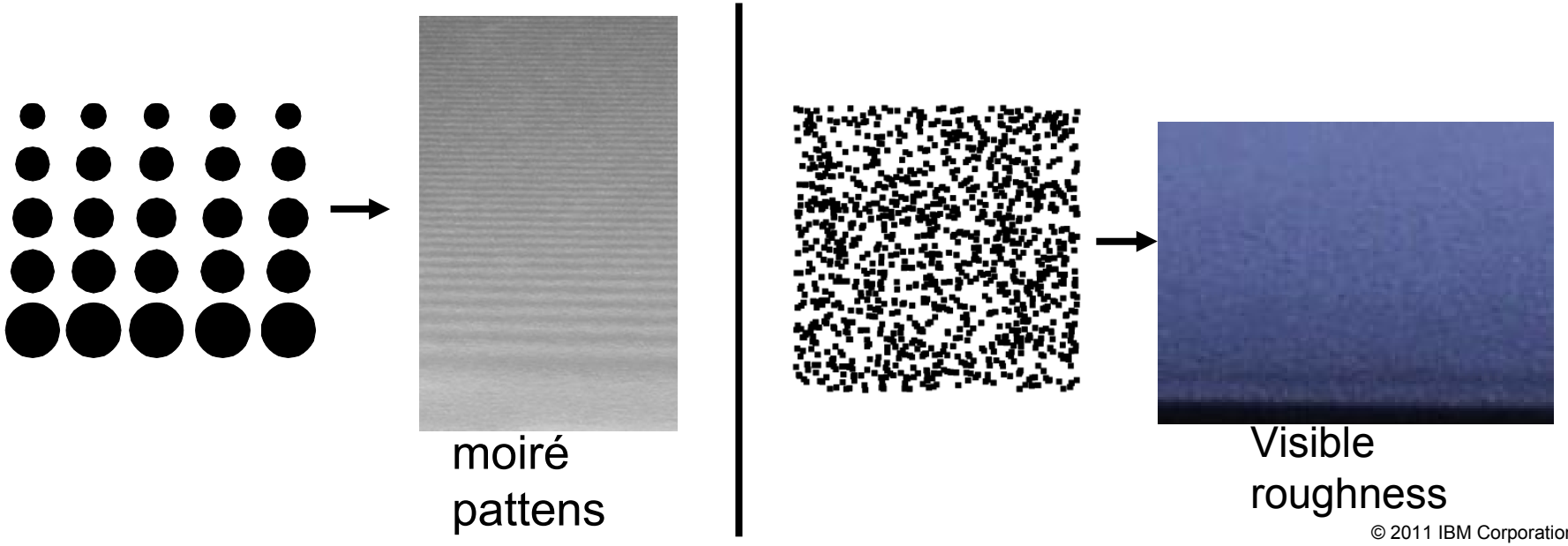
Goal: develop fast method to generate good dot patterns

- Dot pattern generation is a critical step in liquid-crystal display (LCD) manufacturing
  - Used in light gulde and diffuser film for better luminance uniformity
- Present method needs much human efforts
  - Typically, handmade adjustment is needed to re-arrange initial pattern
  - This step takes several days



Requirements of Dot Patterns

- Properly irregular
  - Regular dot distribution causes moiré patterns
- Sufficient uniform
  - Insufficient uniform distribution causes visible roughness
- Capable of providing arbitrary density gradation



Outline

- Motivation
  - Requirements of dot patterns
- Related Work
  - Molecular-dynamics based methods
- Optimization-based Algorithm
  - Unconstrained nonlinear programming problem
- Experiments
- Conclusion

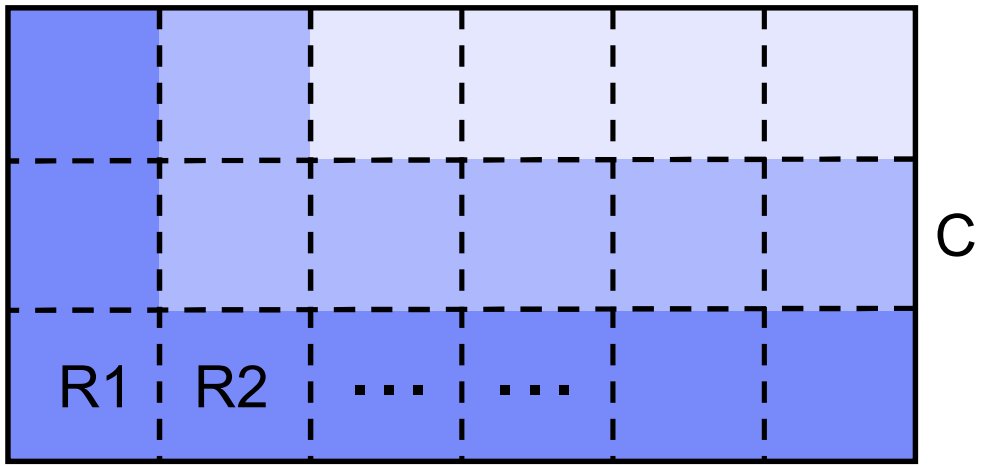
Related work

- Molecular-dynamics-based approaches
  - Fix dot radii and optimize dot positions
  - Define interaction forces between dots and obtain a steady state by solving the equation of motion
    - Idé et al (2003) (see details later)
    - Chang and Lee (2007)
    - Chang, Fang, and Ju (2009)

- Optimization-based approach (Chang and Fang 2007)
  - Fix dot positions and optimize dot radii
  - Optimize the radii of the dots iteratively by computing the luminance output and adjusting the dot radii

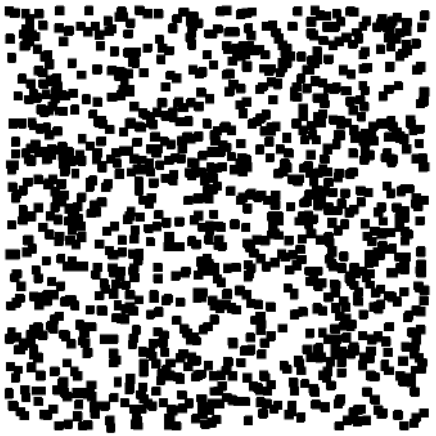
Focus on Optimizing Dot Positions

- Input
  - Container (rectangular): C
  - Dot radius r_0
  - Dot density ρ_i at each grid R_i in the container
    ρ = (area of dots in R) / (area of R)
- Output
  - Dot layout
    - Dot density meets the input density
    - Dot pairs do not overlap

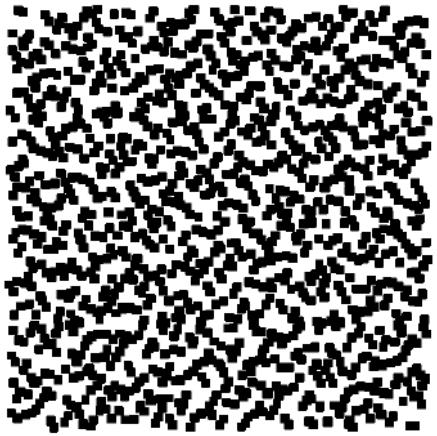


Simulation-based Approach – Step 1 / 2

- Place dots at random using the low discrepancy sequences (LDS) instead of the pseudo-random numbers
  - Use a number of LDS to choose a grid to place a dot
  - Use a pair of numbers of LDS to choose a position in the grid
- Idé et al. (2003) showed that LDS-based initial dot pattern had less visible roughness than pseudo-random-number-based dot patterns



pseudo-random number



LDS

Simulation-based Approach – Step 2 / 2

- Modify the initial dot pattern to make the dot distribution uniform
  - Apply the molecular dynamics simulation by defining interaction forces between dots
- Inter-dot force

$$\boldsymbol{f}_{ij} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} \times \begin{cases} 1 & b_{ij} < D, \\ \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - b_{ij}}{L}\right) & b_{ij} \geq D, \end{cases}$$

- Equation of motion

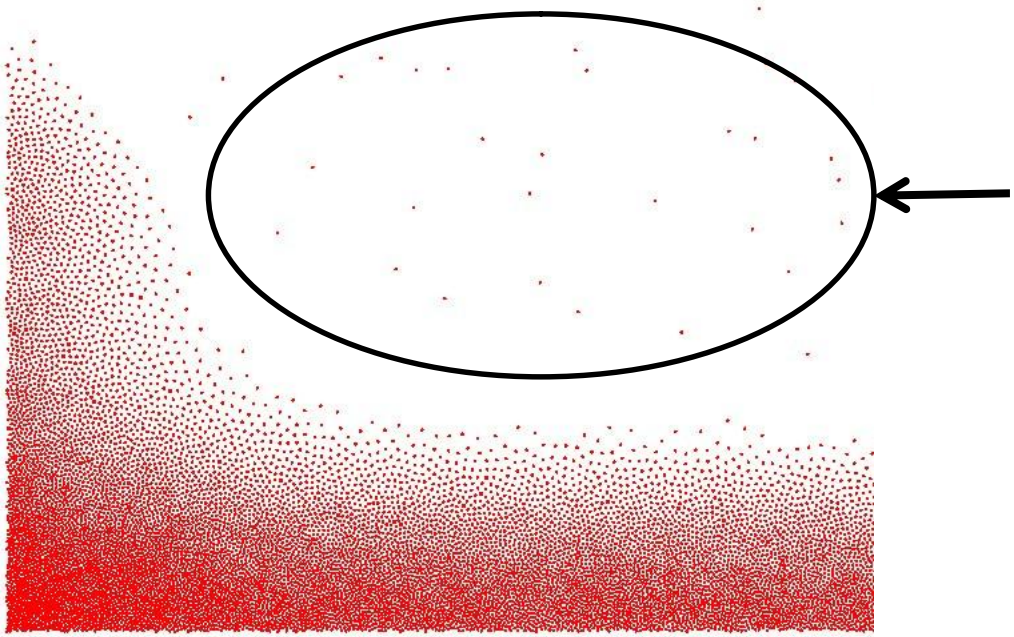
$$m\frac{d^2\boldsymbol{r}_i}{dt^2} + c\frac{d\boldsymbol{r}_i}{dt} = \sum_{j \neq i} \boldsymbol{f}_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j), \quad i = 1, \ldots, n$$

- Differential equation

$$\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \frac{1}{c}\Delta t \sum_{j=1}^{n} \boldsymbol{f}_{ij}(t)$$

Drawbacks of the Algorithm by Idé et al. (2003)

- To solve the differential equation takes a lot of time
  - They need to compute force of vast number of dot pairs
- Some dots move far away from the initial positions



Outline

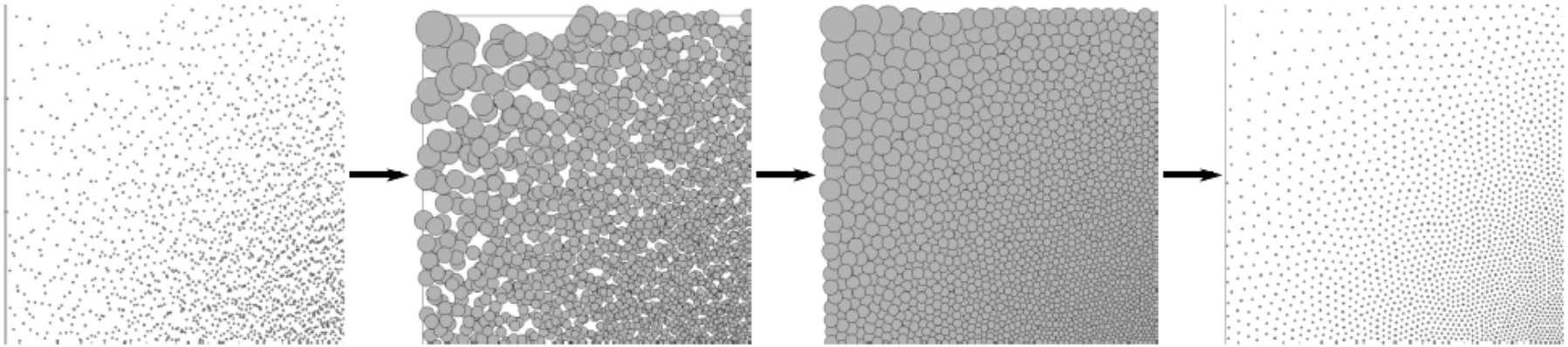
- Motivation
  - Requirements of dot patterns
- Related Work
  - Molecular-dynamics based methods
- Optimization-based Algorithm
  - Unconstrained nonlinear programming problem
- Experiments
- Conclusion

Outline of Our Optimization-based Approach

OptDot

1. Generate an initial dot pattern by LDS (same as Idé et al.)
2. Replace the dots by circles with radii computed from the density
3. Remove the overlaps of circles
4. Replace the circles by dots

New



1 2 3 4

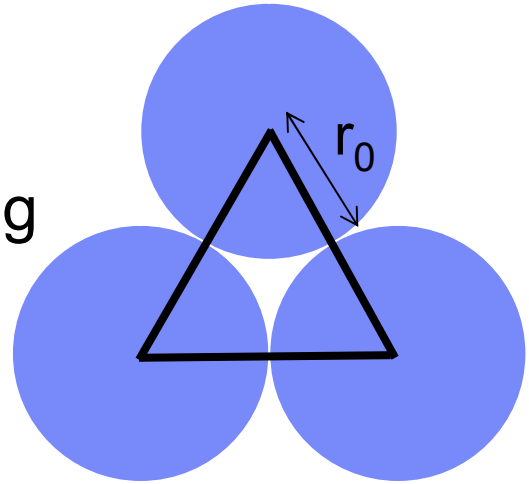
Replace the Dots by Circles

- Each dot is replaced by a circle
  - Equalize the distances between dots by removing the overlaps between circles
  - Design the radii so that circles touch each other after overlap removal
  - Determine the radii from the density
    - Dense region ⇒ the radius should be small
    - Sparse region ⇒ the radius should be large
- Notation: r_0 = dot radius, ρ = density
- The radius of the circles in the dense packing

$$r' = r_0 \sqrt{\frac{\pi}{2\sqrt{3}\rho}}$$

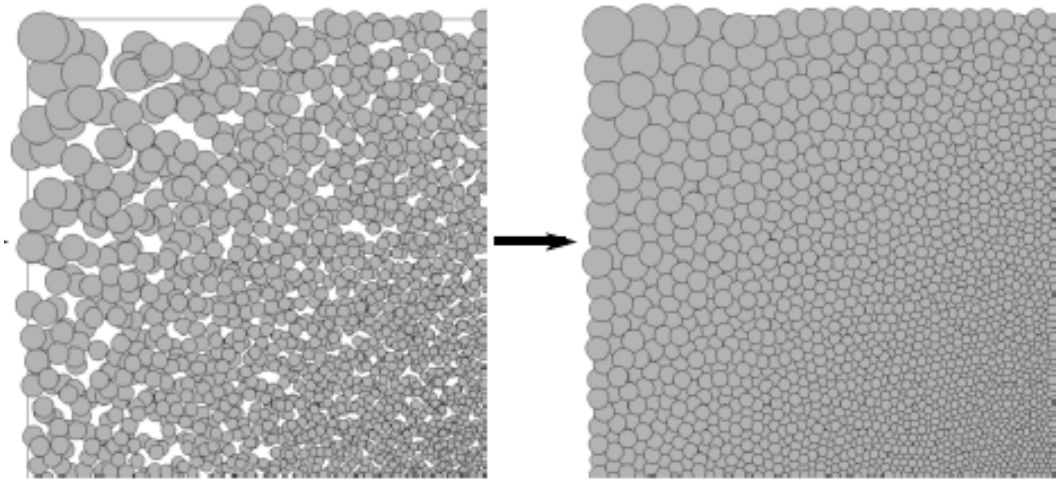
- We use slightly larger radius to strengthen the equalization effect

$$r = r_0 \sqrt{\frac{\pi}{\sqrt{3}\rho}}$$



Remove the Overlap of Circles

- We define three types of penalties and minimize them to perturb the dot pattern
  - Penalty of overlap between circles
    - To equalize the inter-dot distances
  - Penalty of protrusion of circles from the container
    - To include all dots within the container
  - Penalty of protrusion of each circle from a region assigned to each circle
    - To prevent each circle from moving far away from the initial position

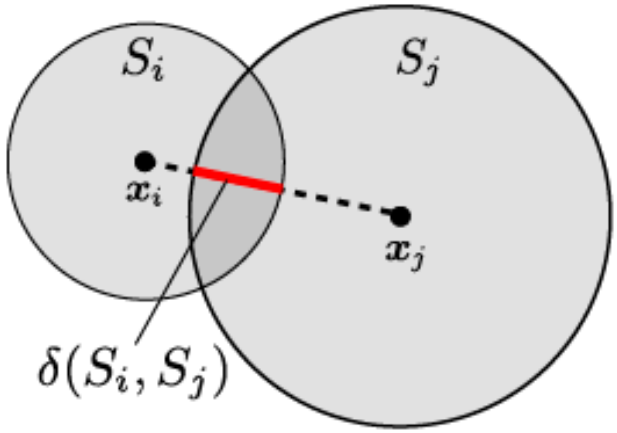


Penalty of Overlap Between Circles

- Penetration depth (Dobkin et al. 1993)
  - Minimum translation distance to remove the overlap of the shapes
  - Easy to derive for circles

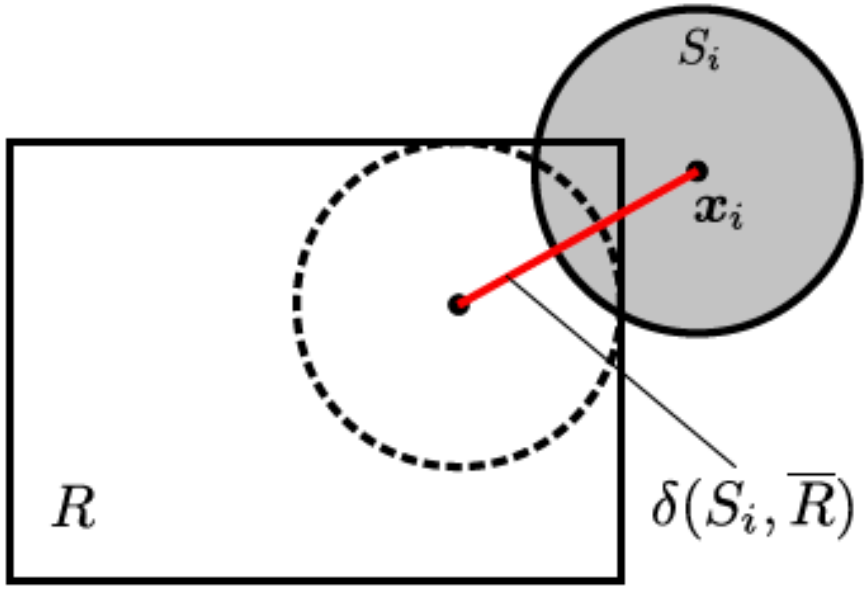
$$\delta(S_i, S_j) = \max\{r_i + r_j - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|, 0\}$$

- We adopt the squared penetration depth as the penalty of overlap
  - The squared case is differential
  - The derivative of the squared case is simple



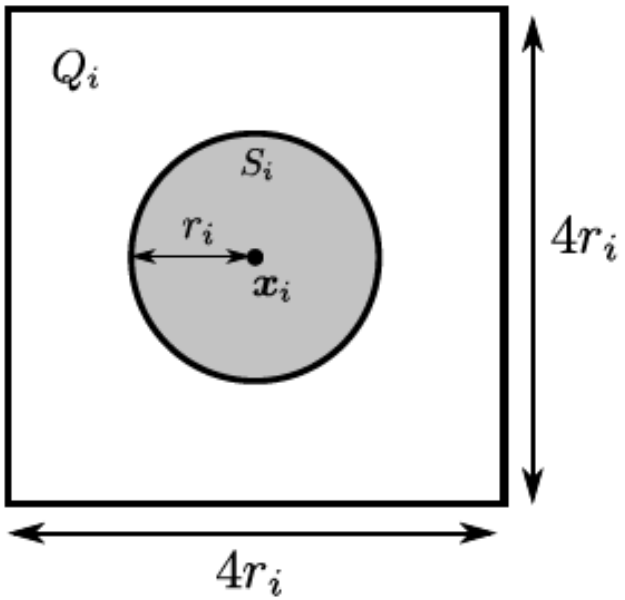
Penalty of Protrusion of Circles From the Container

- We adopt the penalty of protrusion of circles from the container as the squared penetration depth of the circle and the complement of the container



Motion Restriction for Each Circle

- We assign a square region Qi to each circle Si
  - We set the edge length by twice the diameter of the circle
- We adopt the squared penetration depth of Si and the complement of Qi to restrict the motion of the circle



Formulation of Overlap Removal

- Inputs
  - Container: R, Complement of the Container $\overline{R}$
  - Circles: $\{S_i = ((x_i, y_i), r_i) \mid i = 1, \ldots, n\}$
  - Domain for each circle: $\{Q_i \mid i = 1, \ldots, n\}$
- Decision variables
  - Circle positions: $\{\boldsymbol{x}_i \in \mathbb{R}^2 \mid i = 1, \ldots, n\}$
- Formulation

penetration of circles

$$\text{minimize} \quad \sum_{1 \le i < j \le n} \delta(S_i \oplus \boldsymbol{x}_i, S_j \oplus \boldsymbol{x}_j)^2$$

protrusion from the container　　protrusion from the restricted region

$$+ \sum_{1 \le i \le n} \delta(S_i \oplus \boldsymbol{x}_i, \overline{R})^2 + \sum_{1 \le i \le n} \delta(S_i \oplus \boldsymbol{x}_i, \overline{Q_i})^2$$

$$\text{subject to} \quad \boldsymbol{x}_i \in \mathbb{R}^2, \quad i = 1, \ldots, n.$$

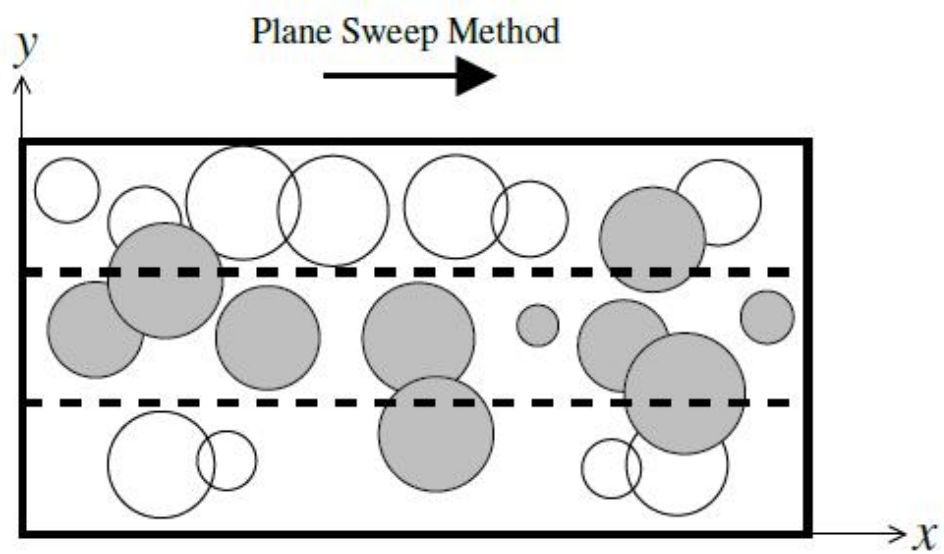
How to Solve the Optimization Formulation

- The formulation is unconstrained nonlinear optimization problem
  - The objective function is differential

- Apply the limited memory BFGS (L-BFGS) method (Liu and Nocedal 1989)
  - Iteratively update the variables using the gradient and an approximate inverse Hessian
  - L-BFGS can be applied to large-scale problems because it requires O(n) space
    - In our problem instances, there are >100k variables
  - L-BFGS obtains the locally optimal solution quickly in practice

Evaluation of the Objective Function and the Derivative

- Evaluation of the objective function and the derivative
  - Penalty of overlap between circles
    - Apply the plane sweep method by Imamichi and Nagamochi (2008)
    - O(n log n + K) time where K is the number of overlapping pairs (Note that K is usually O(n) in our algorithm because we set the radii of circles to touch each other)
  - Penalty of protrusion of circles from both the container and the restricted region
    - O(n) time



Relation between Optimization and Simulation

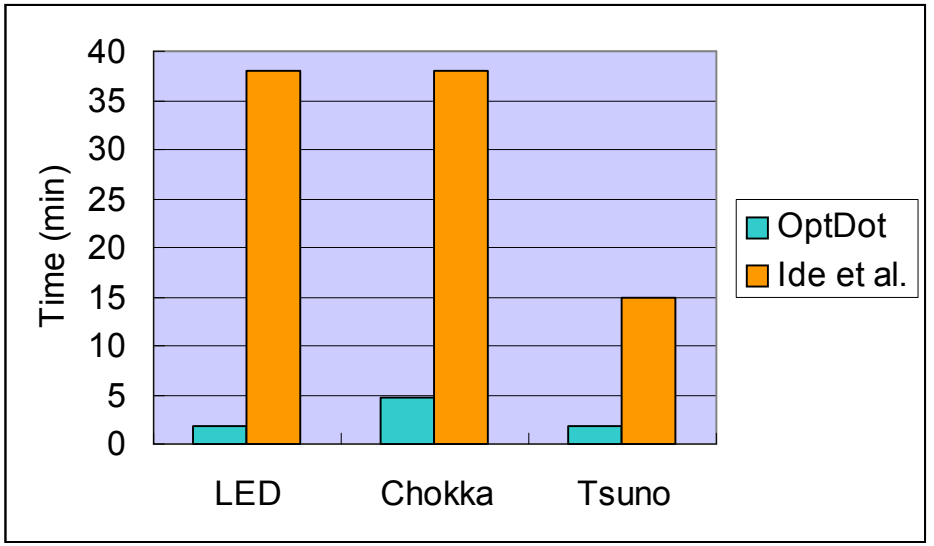
- The optimization formulation gives a macroscopic insight into the microscopic molecular dynamics simulation
  - Penetration penalty in optimization formulation corresponds to inter-dot force in simulation
  - The (locally) optimal solution in the optimization formulation corresponds to the steady state in the simulation
- The optimization-based approach obtains a (locally) optimal solution faster than the simulation-based approach
  - Simulation-based methods are suitable for tracing the changes of the system for entire periods of time
  - Optimization-based methods are designed to quickly converge to a (locally) optimal solution

Outline

- Motivation
  - Requirements of dot patterns
- Related Work
  - Molecular-dynamics based methods
- Optimization-based Algorithm
  - Unconstrained nonlinear programming problem
- Experiments
- Conclusion

Computational Experiments

- Three Instances (Dot diameter = 46 μm)
  - Tsuno : 111211 dots
  - Chokka : 287080 dots
  - LED : 117088 dots
- Environment
  - Run on a PC with Core 2 Duo T9300 CPU (2.5GHz)
  - OptDot is implemented in C++
- Comparison of running time
  - OptDot runs much faster than the algorithm by Idé et al.



Comparison of Number of Close Dot Pairs

- Our algorithm succeeded to obtain better dot patterns for all instances
  - The number of colliding or too close dot pairs is smaller

Table 1. Number of colliding dot pairs, i.e., the distance < 46μm

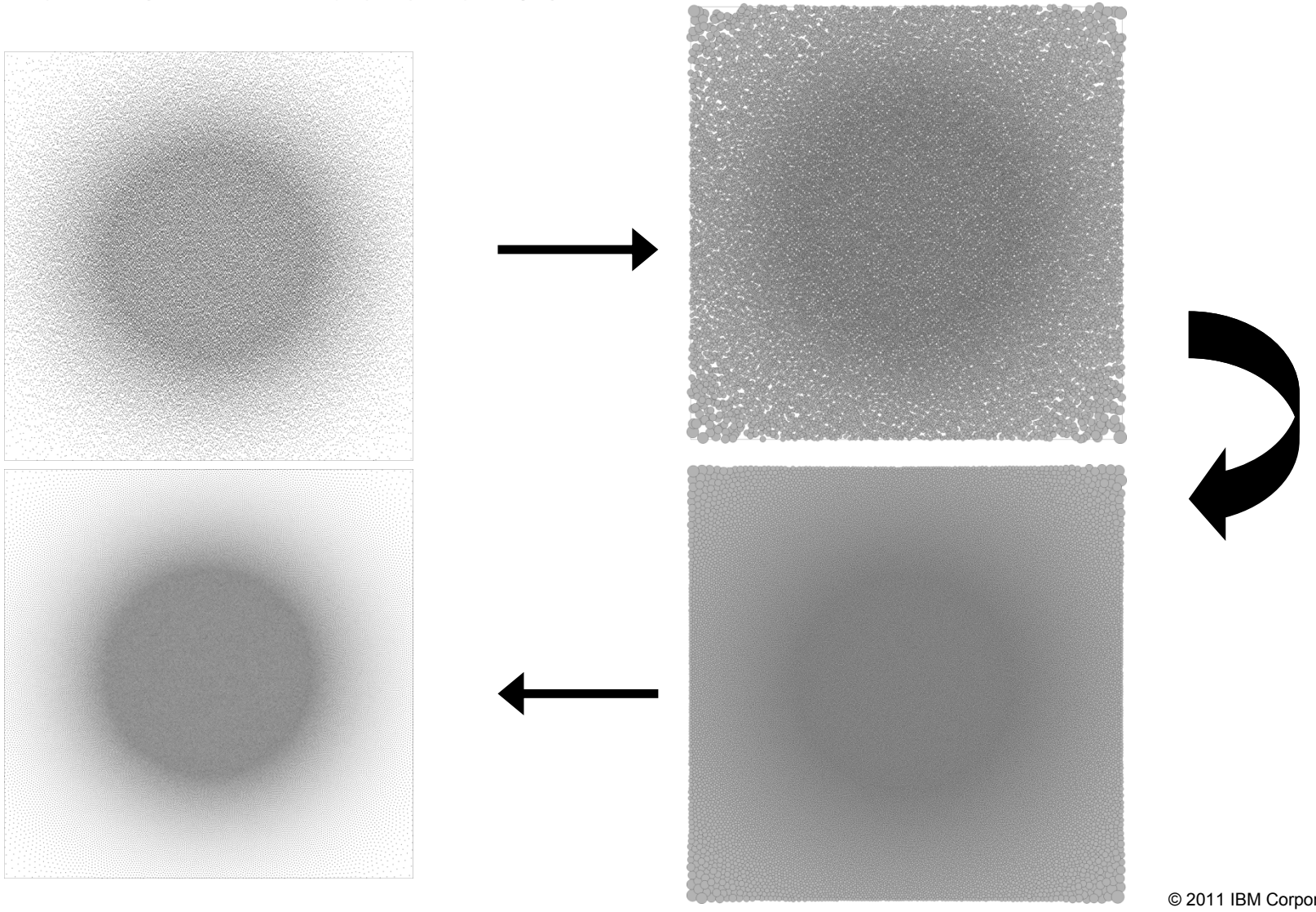
Pattern	Initial layout	Idé et al.	OptDot	# dots	# dot pairs
Tsuno	24339	1	0	111211	6183887655
Chokka	36827	0	0	287080	41207319660
LED	57949	1315	0	117088	6854741328

Table 2. Number of too close dot pairs, i.e., the distance < 60μm

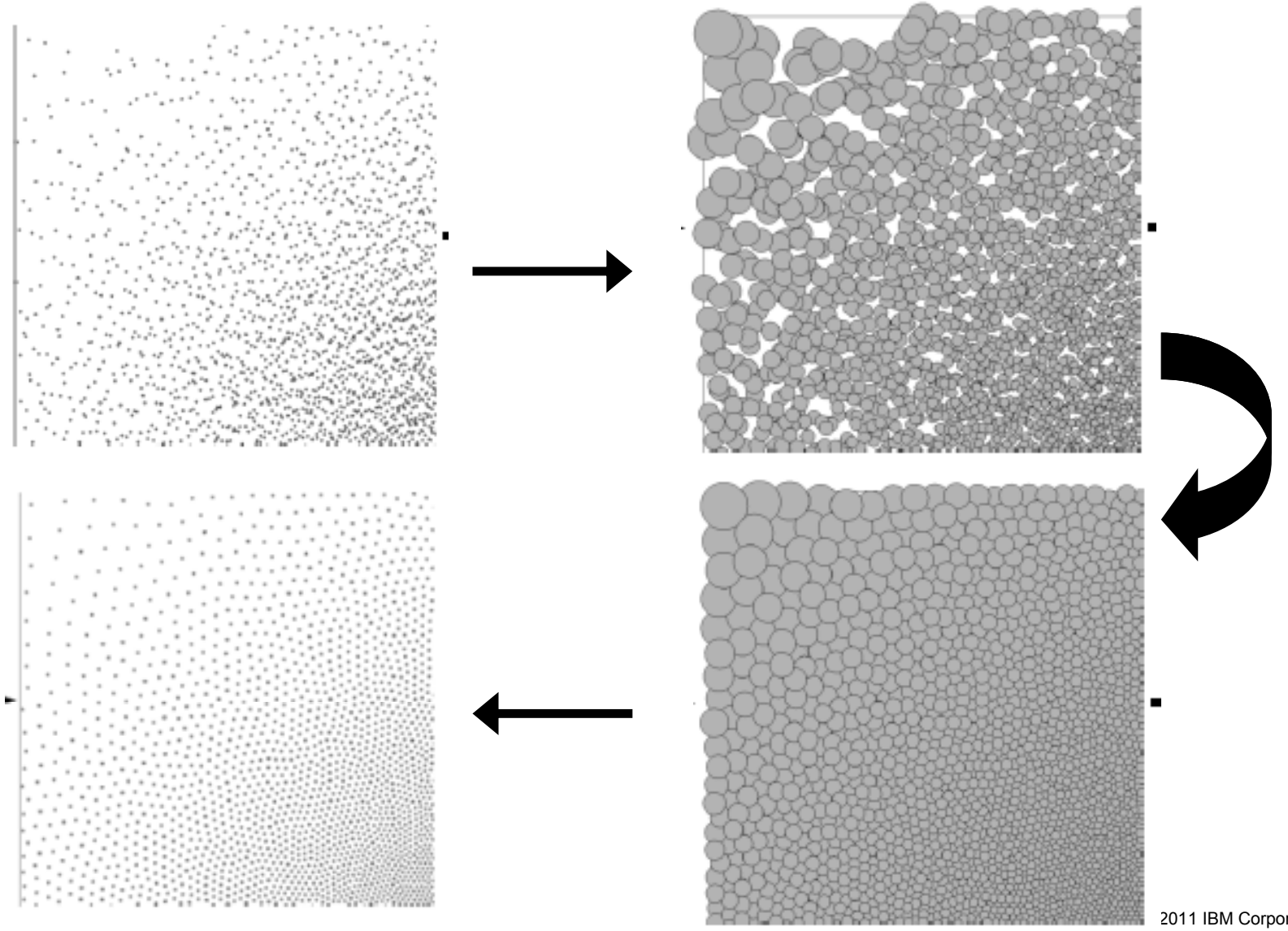
Pattern	Initial layout	Idé et al.	OptDot	# dots	# dot pairs
Tsuno	45523	66	16	111211	6183887655
Chokka	73055	2	0	287080	41207319660
LED	104530	37844	32644	117088	6854741328

Results (LED, #dots = 110k)

- OptDot 1.8min
- Idé et al 38min

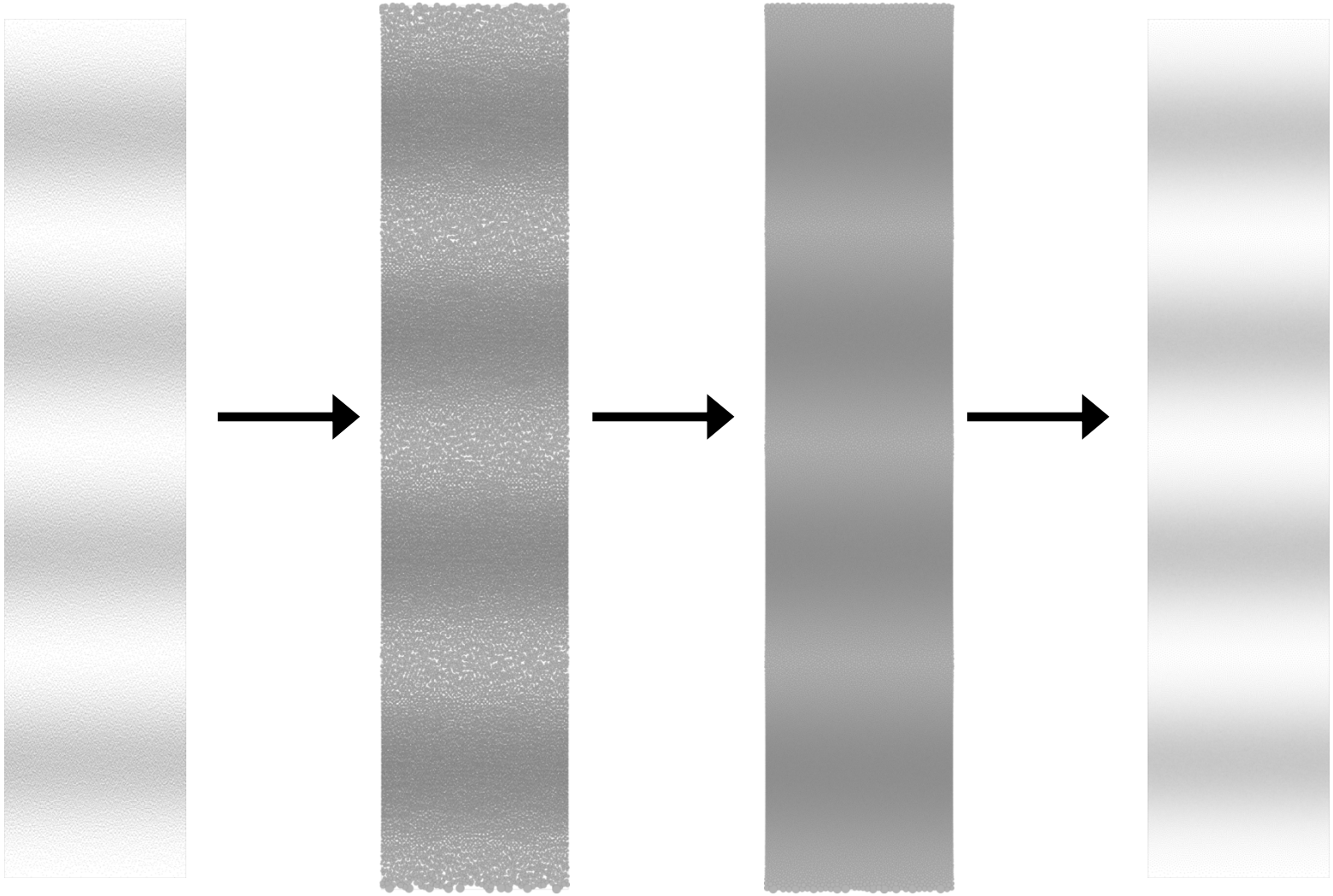


Results (LED, #dots = 110k), Magnified Pictures

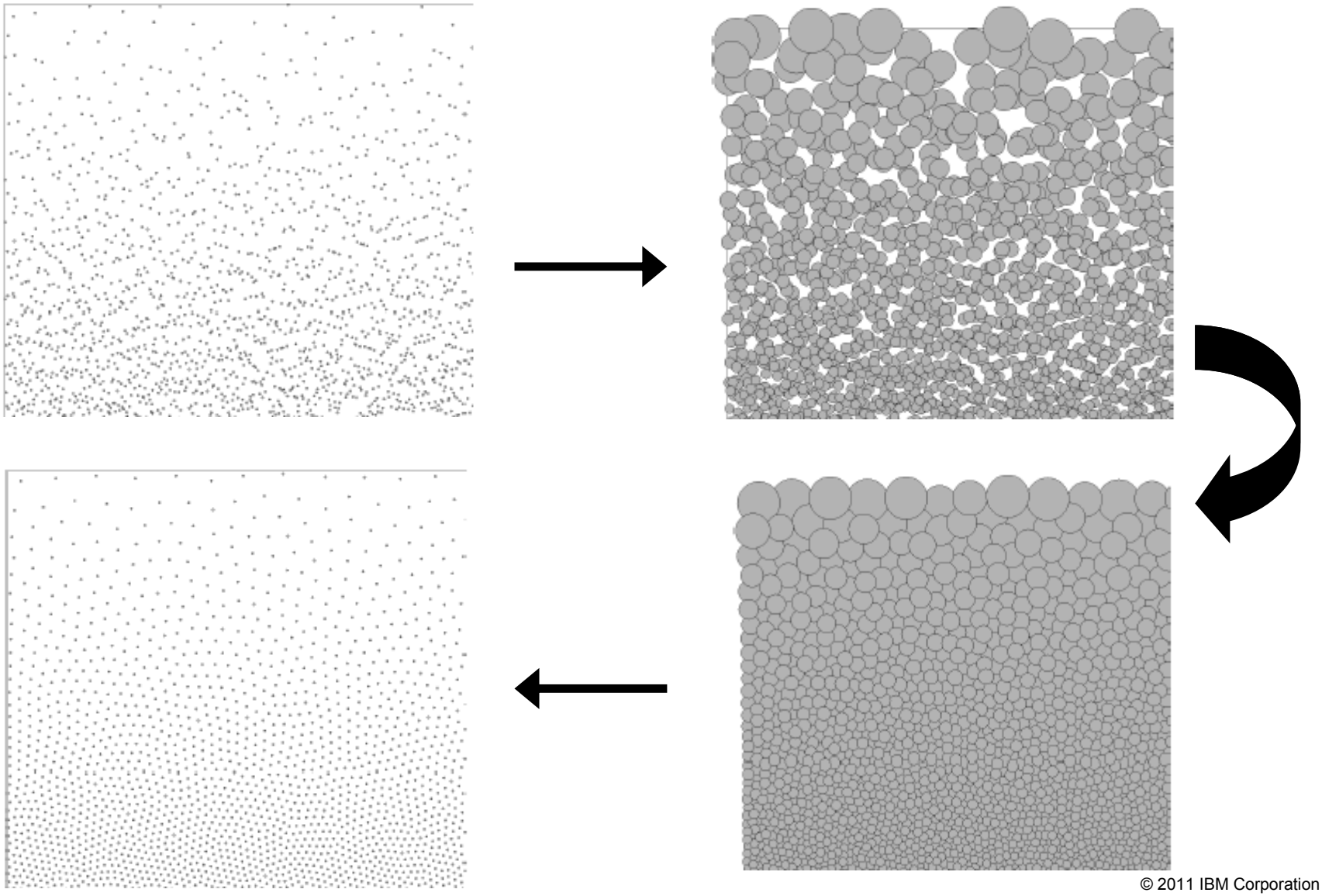


Results (Chokka, #dots = 280k)

- OptDot 4.7min
- Idé et al. 38min

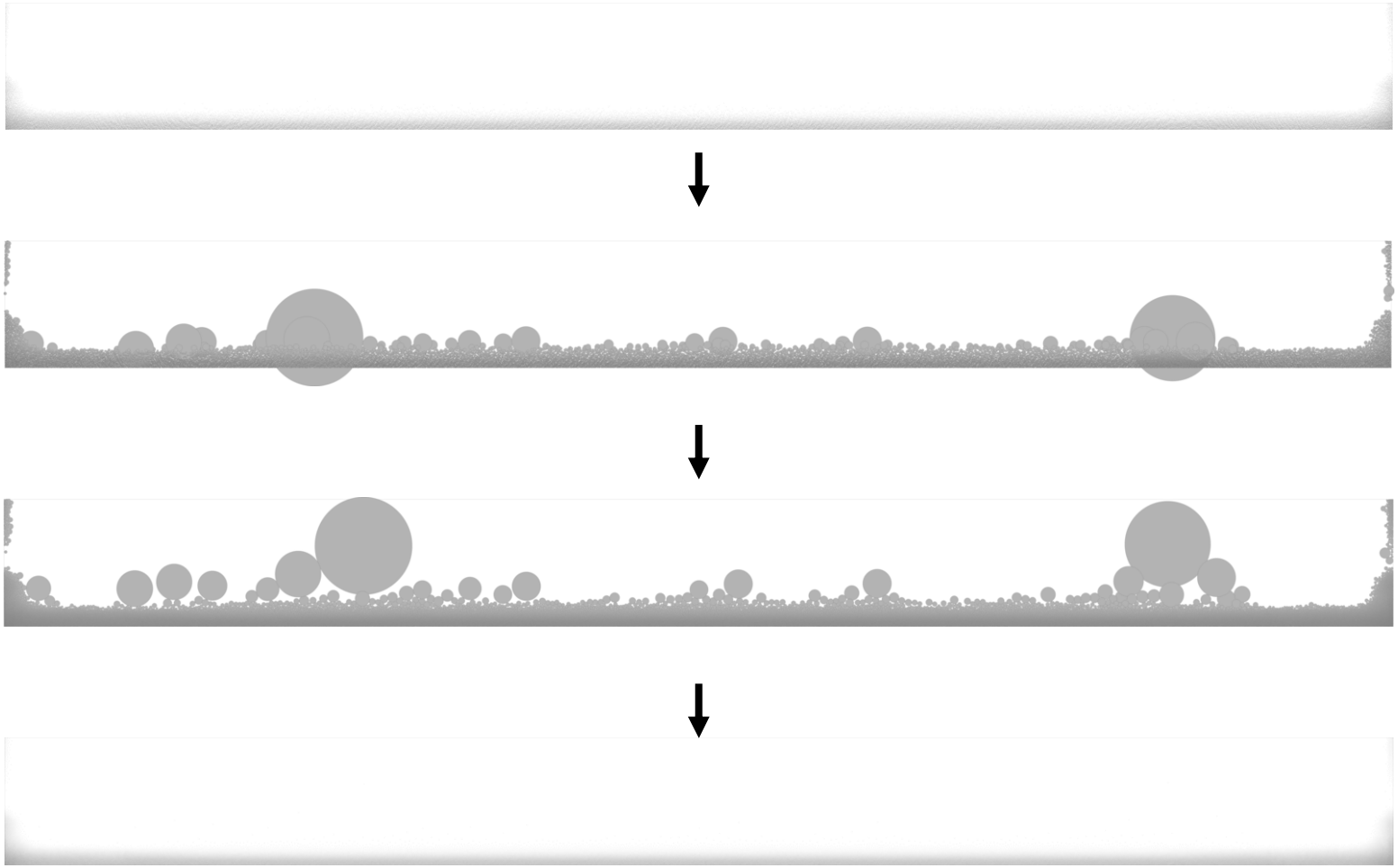


Results (Chokka, #dots = 280k), Magnified Pictures

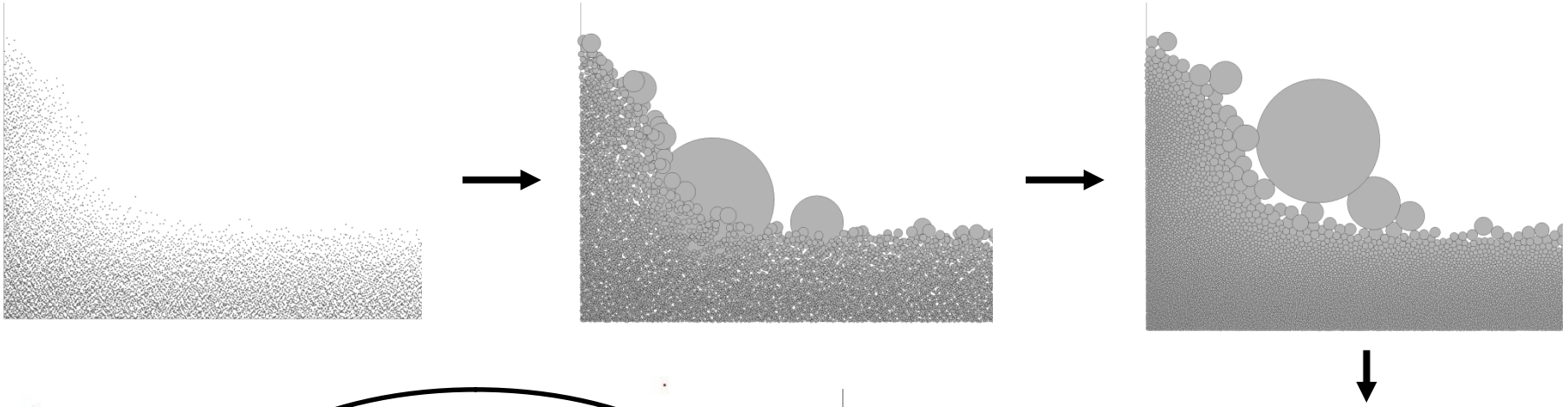


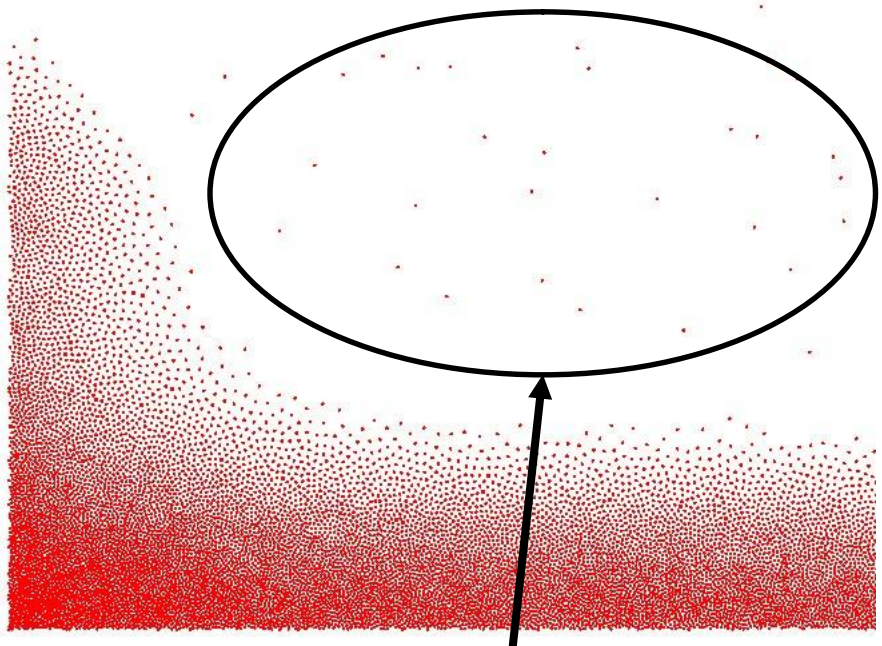
Results (Tsuno, #dots = 110k)

- OptDot 1.9min
- Idé et al. 15min



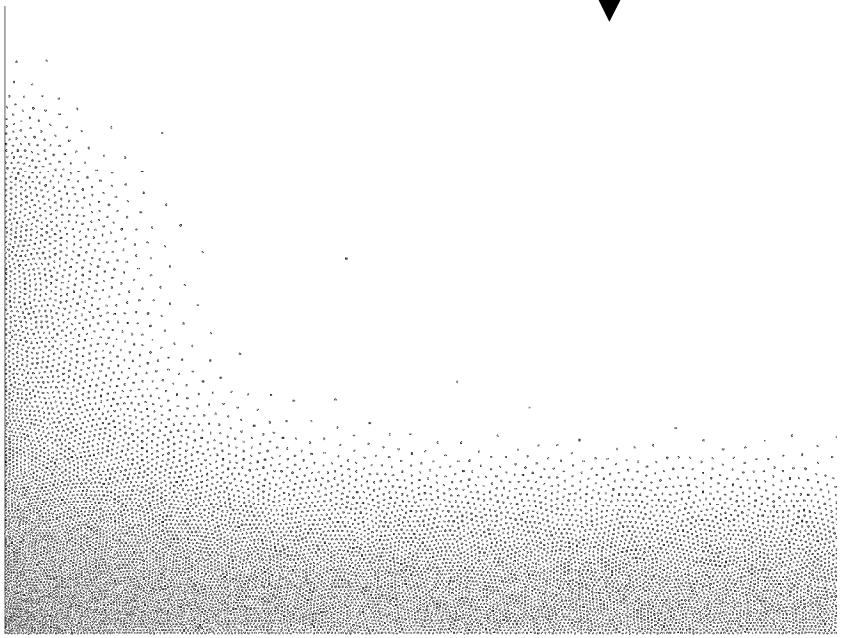
Results (Tsuno, #dots = 110k), Magnified Pictures





Idé et al.

Some dots moves here



OptDot

Outline

- Motivation
  - Requirements of dot patterns
- Related Work
  - Molecular-dynamics based methods
- Optimization-based Algorithm
  - Unconstrained nonlinear programming problem
- Experiments
- Conclusion

Conclusion

- We propose an optimization-based dot pattern generation algorithm
  - Converged to a locally optimal solution faster than the simulation-based method
  - The solution quality is comparable to that by the simulation-based method
  - Optimization-based method is suitable to problems that focus only on the locally optimal solution or steady state
- Future work
  - Extend the optimization-based method to non-circular dot patterns such as ellipses or rectangles
  - Evaluate the physical characteristics of the optical components with the dot patterns generated by our method