Practical Virtual Coordinates for Large Wireless Sensor Networks

Jiangwei Zhou*, Yu Chen+,
Ben Leong∇, Boqin Feng*
**Xi'an Jiaotong University*
+ Duke University
∇ National University of Singapore

Practical Virtual Coordinates for Large Wireless Sensor Networks

Jiangwei Zhou*, Yu Chen+,
Ben Leong∇, Boqin Feng*
*Xi'an Jiaotong University
+ Duke University
∇ National University of Singapore

Wireless Sensor Networks:

Point-to-point routing is a useful primitive for data-centric applications

For small sensor networks (<200 nodes): pick your favorite algorithm. ☺

SenSys 2010

For large sensor networks (~3,200 nodes), geographic routing algorithms are most scalable:

- storage cost proportional to network density not to size
- motes have small RAM

Problem

Geographic routing algorithms require coordinates!

Large networks
⇒ Big problem!

Wait A Moment… .

Hasn't the problem already been solved?

Virtual Coordinates

[Rao et al., Mobicom 2003]

Related Work

- Virtual Coordinates
  - NoGeo (Rao et al., Mobicom 2003)
  - GSpring (Leong et al., ICNP 2007)
  - GLoVE (Westphal et al., Infocom 2009)
- Look good in simulation
- Not practical or efficient for real TinyOS motes
  - 48 KB RAM
  - CC2420: 127-byte messages

Story of our failure

Tried to implement existing algorithms in TinyOS but failed!

Our Solution

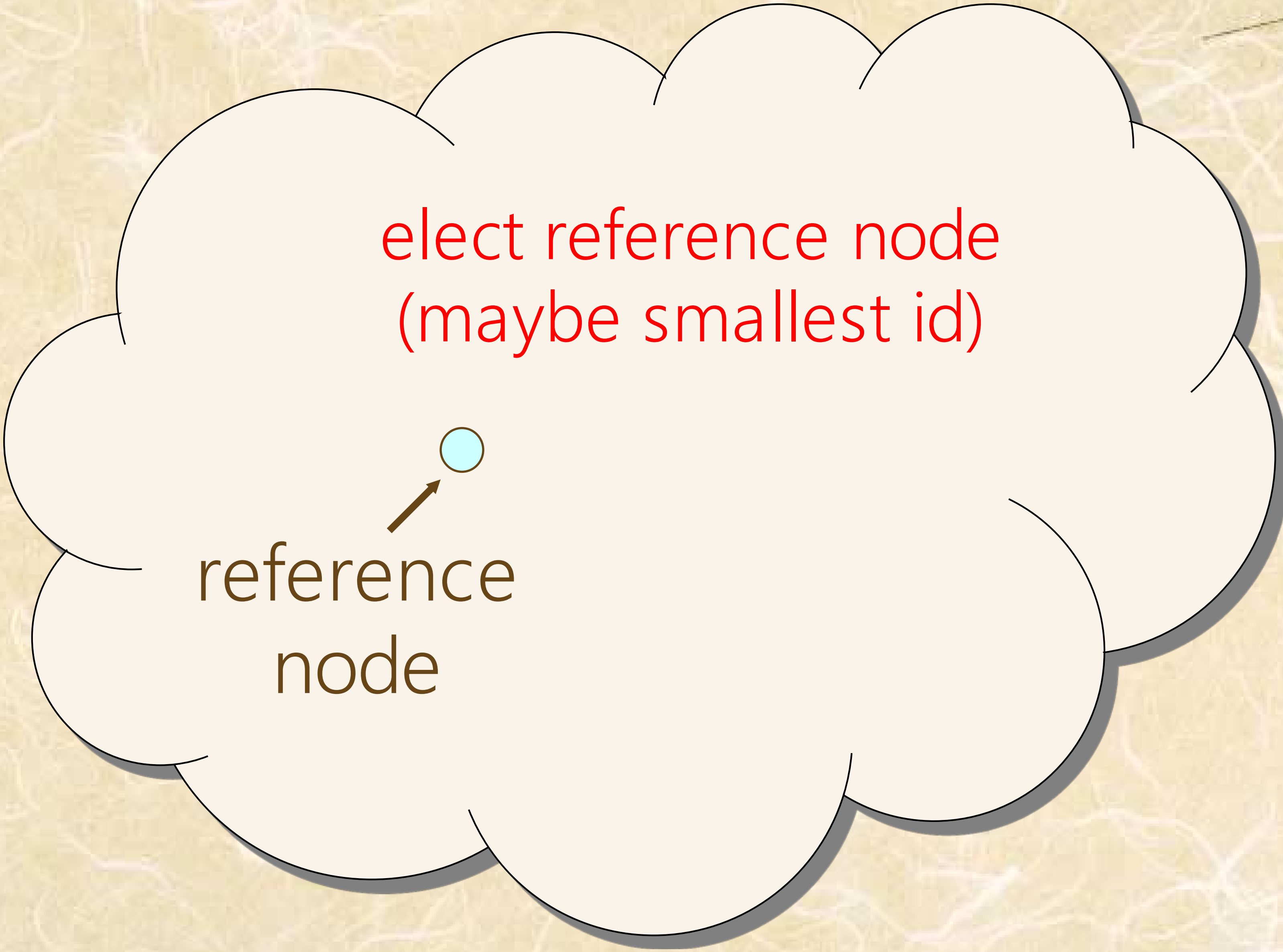
Particle Swarm Virtual Coordinates (PSVC)

Goal: Compute Euclidean Coordinates in a **Scalable & Distributed** Way

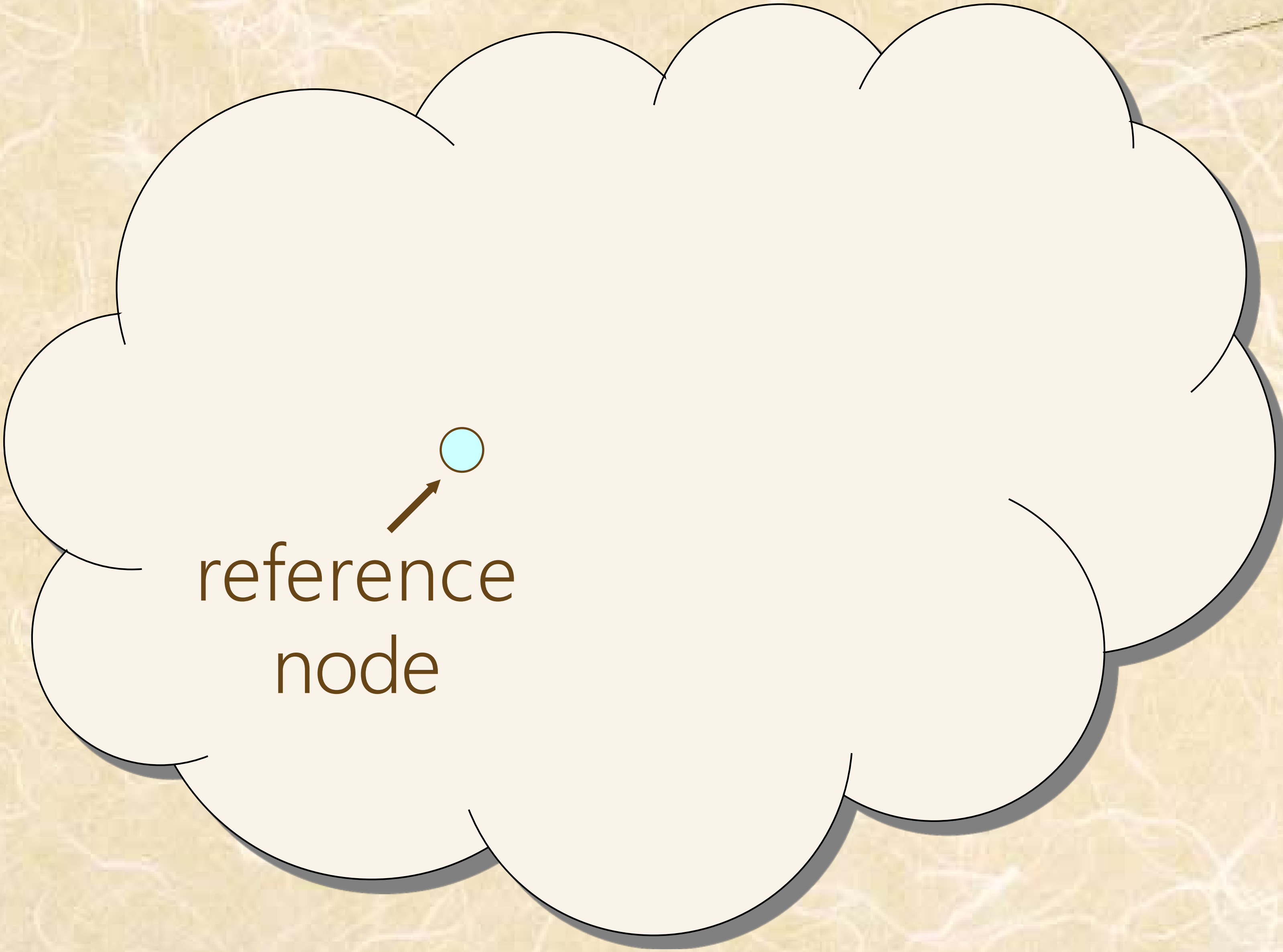
Overview

1. How PSVC works
   - How we fix NoGeo (Rao et. al)
   - Why GSpring (Leong et al.) failed
2. How well PSVC works
   - Performance
   - Cost
3. Current Limitations/Future Work

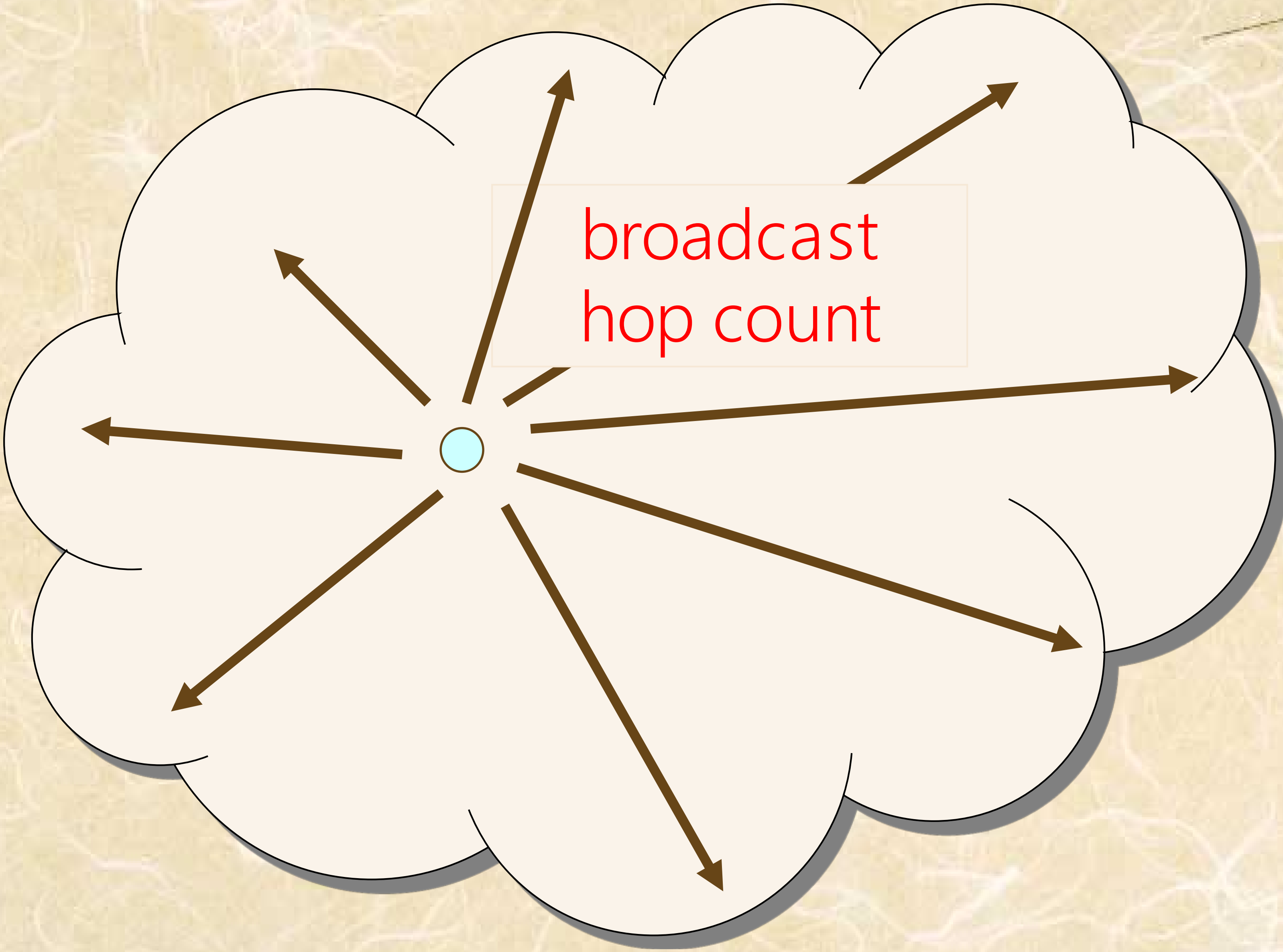
NoGeo (Rao et al., Mobicom'03)



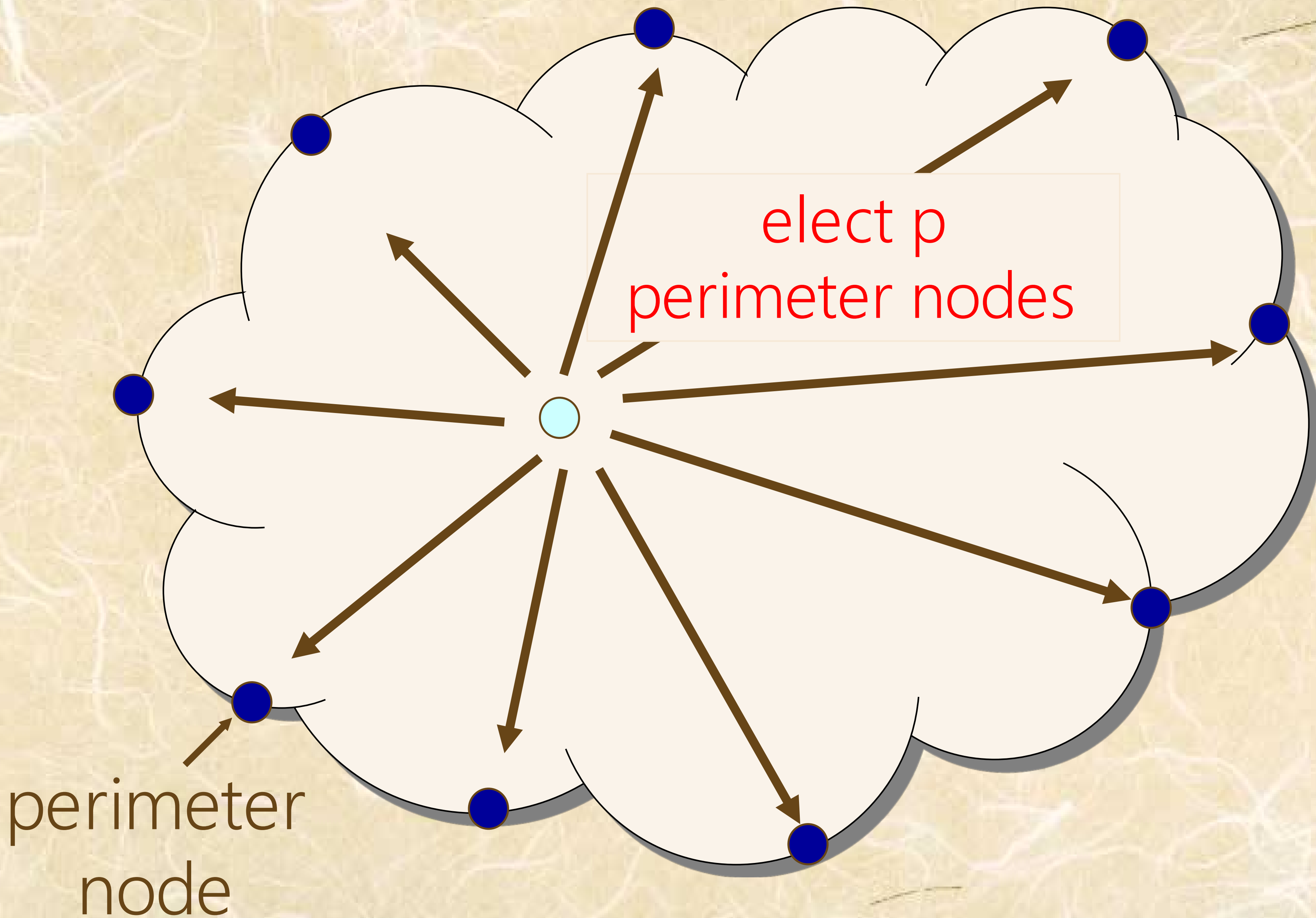
NoGeo (Rao et al., Mobicom'03)



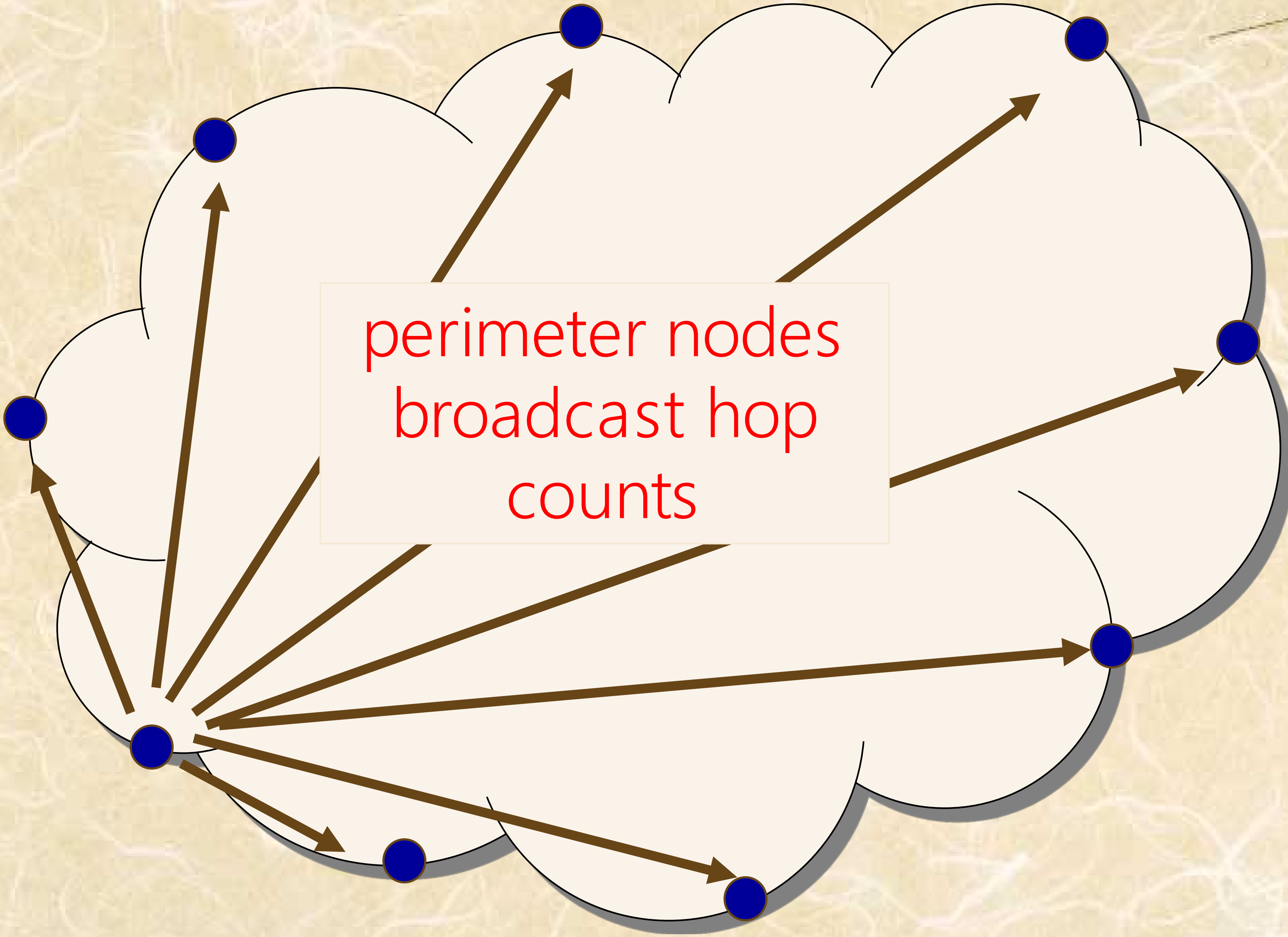
NoGeo (Rao et al., Mobicom'03)



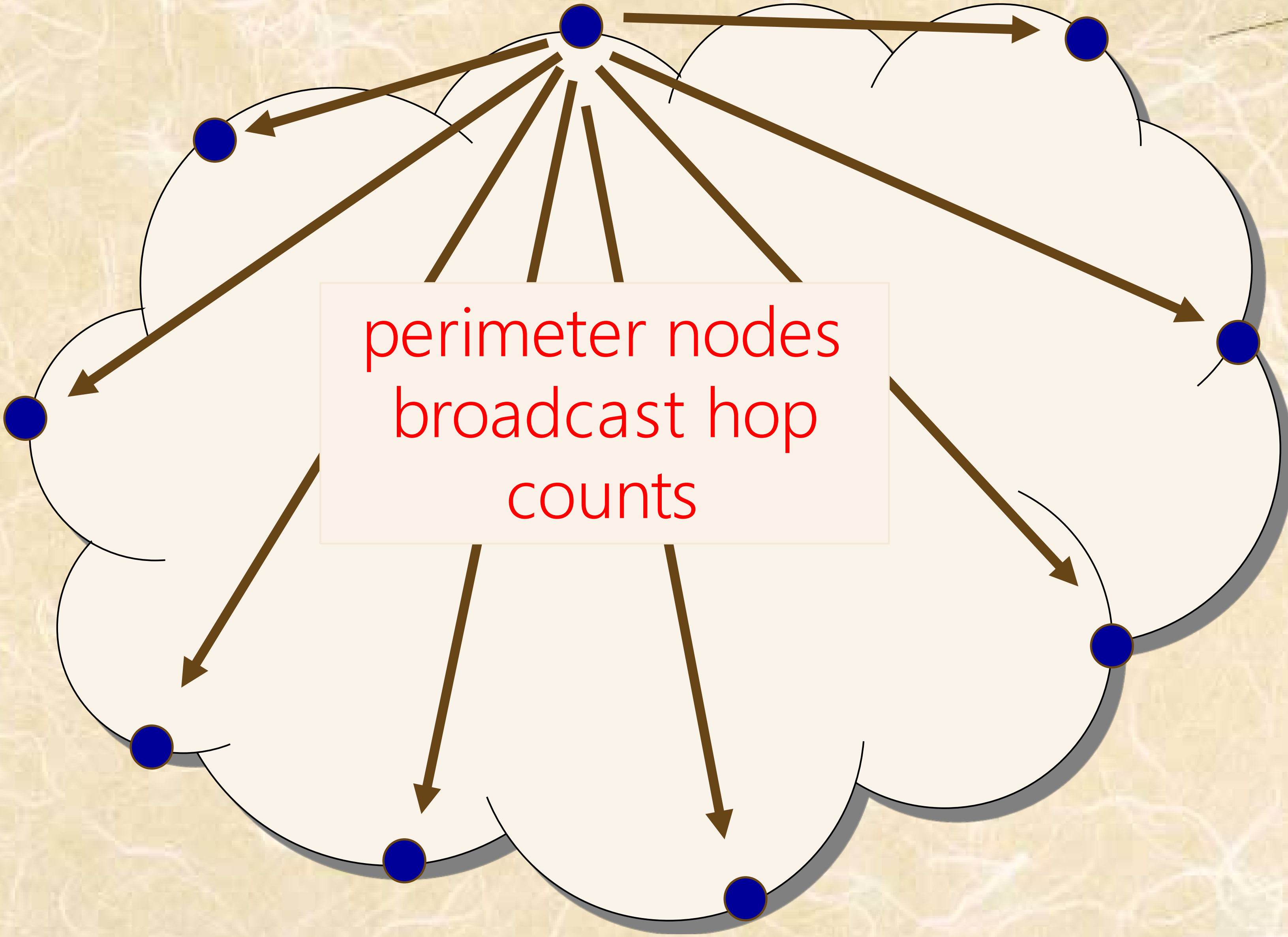
NoGeo (Rao et al., Mobicom'03)



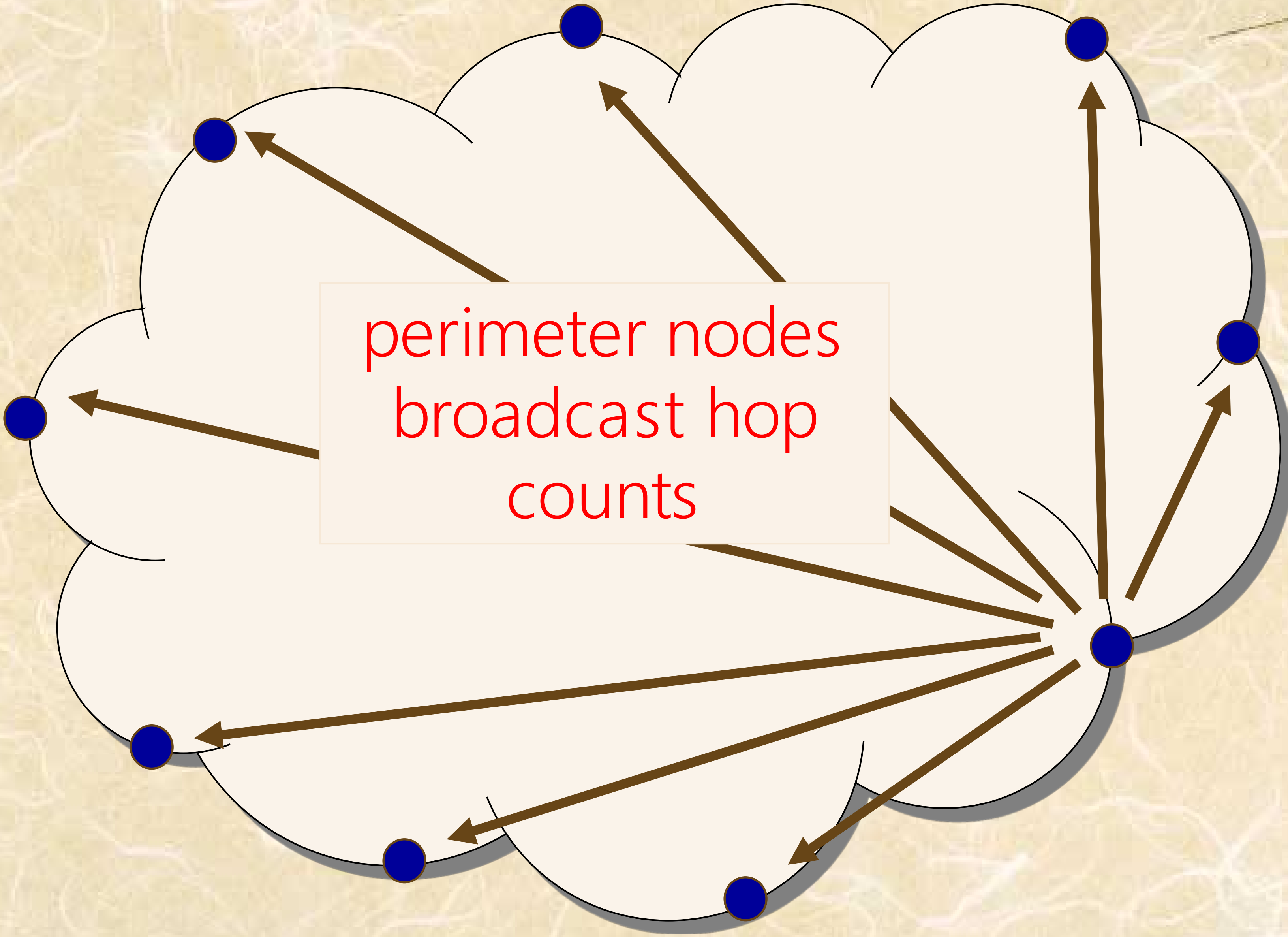
NoGeo (Rao et al., Mobicom'03)



NoGeo (Rao et al., Mobicom'03)



NoGeo (Rao et al., Mobicom'03)



Hop Counts → Coordinates

$$\begin{pmatrix} 0 & h_{12} & \cdots & h_{1p} \\ h_{12} & 0 & \cdots & \vdots \\ \vdots & \vdots & 0 & h_{p-1p} \\ h_{1p} & \cdots & h_{p-1p} & 0 \end{pmatrix} \Longrightarrow (\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_p)$$

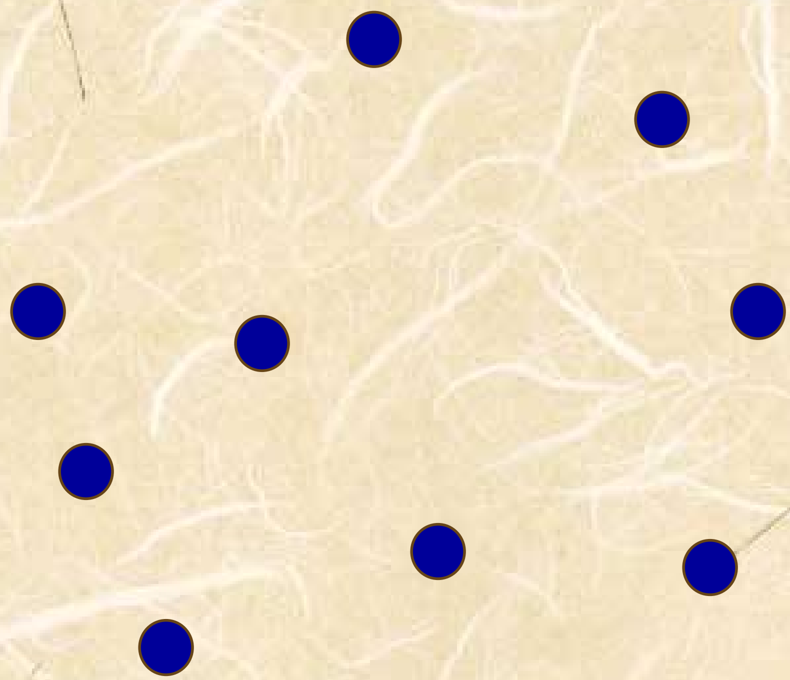
p rows

p columns

$$\min E = \sum_{i=1}^{p} \sum_{j=1}^{p} (|\vec{x}_i - \vec{x}_j| - h_{ij})^2$$

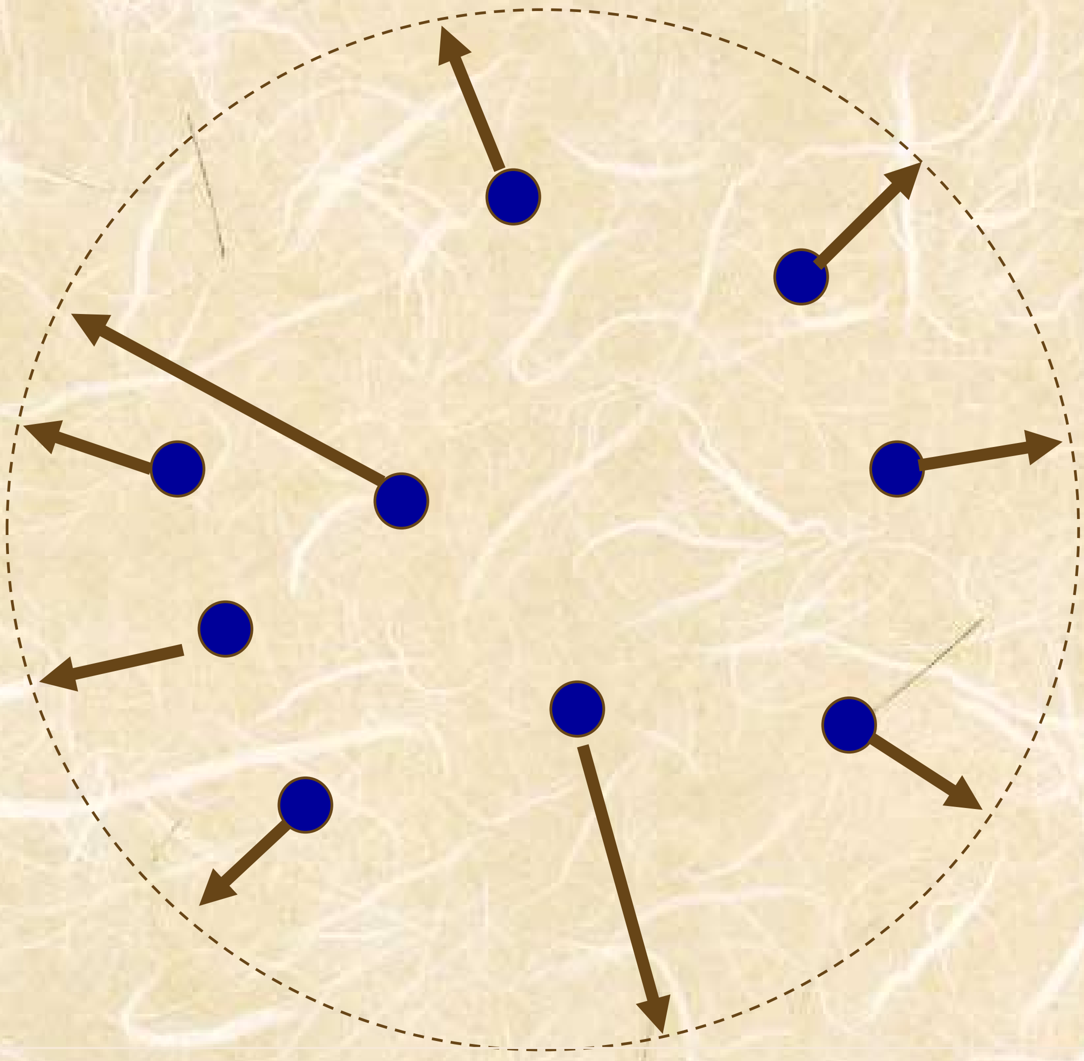
Standard optimization problem (solved numerically)

NoGeo (Rao et al., Mobicom'03)



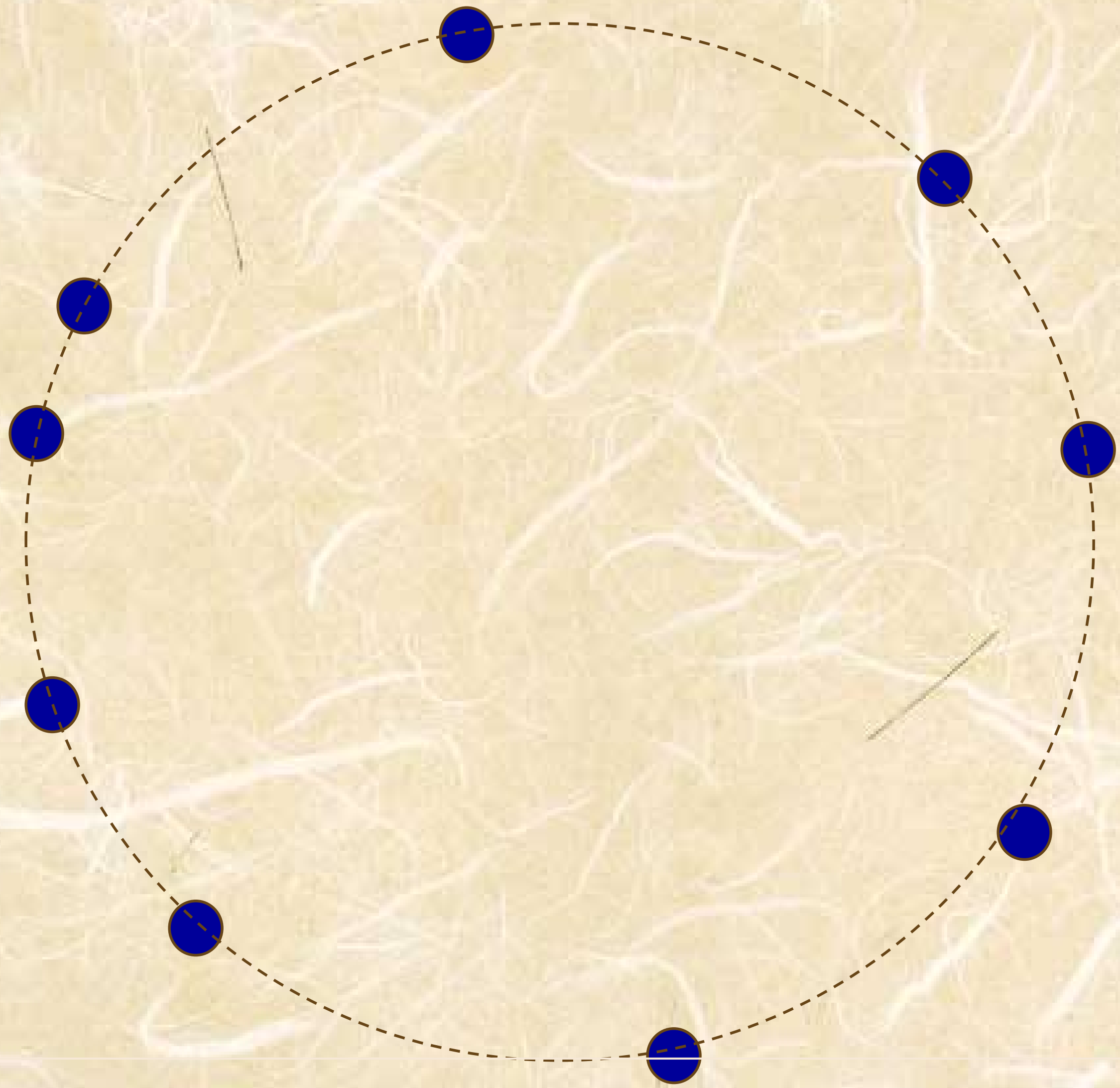
Virtual Coordinate Space

NoGeo (Rao et al., Mobicom'03)



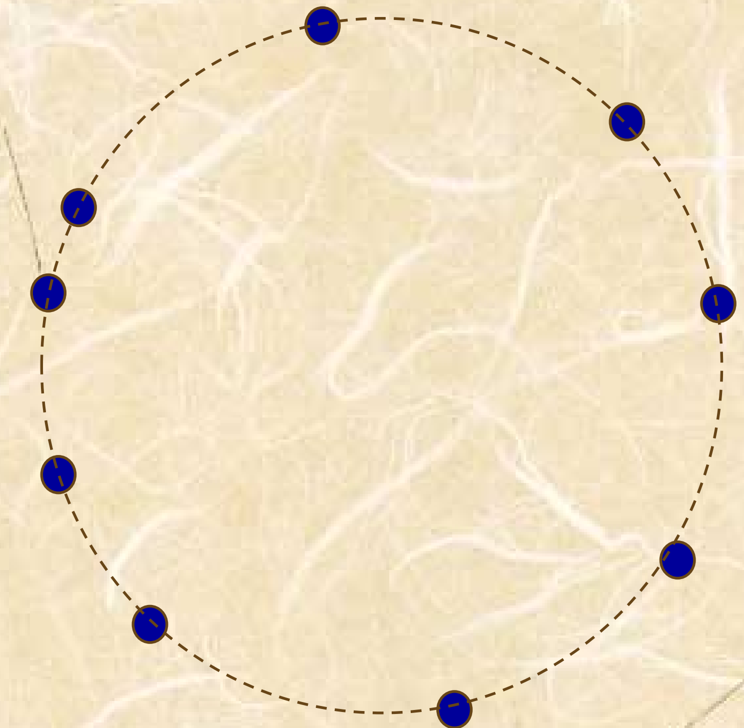
Project onto virtual circle

NoGeo (Rao et al., Mobicom'03)

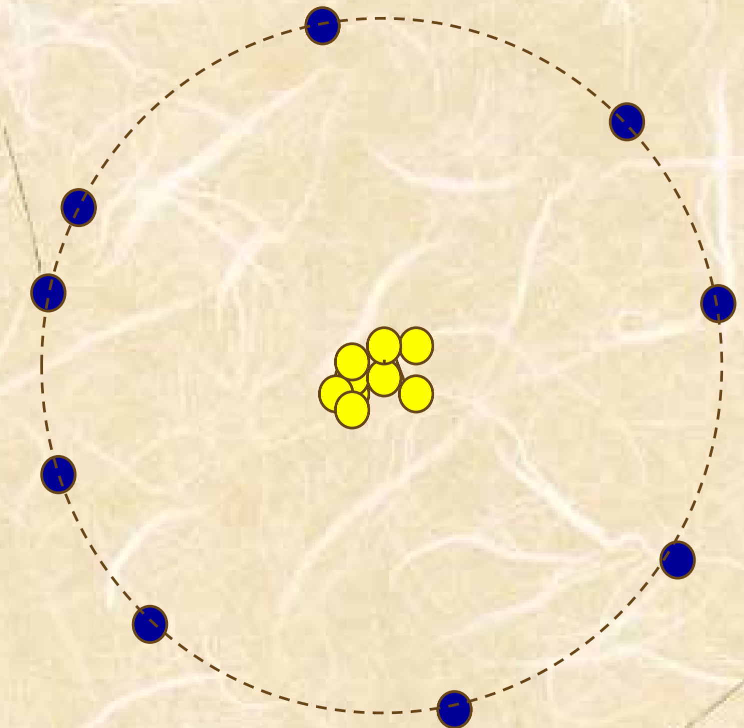


Project onto virtual circle

NoGeo (Rao et al., Mobicom'03)

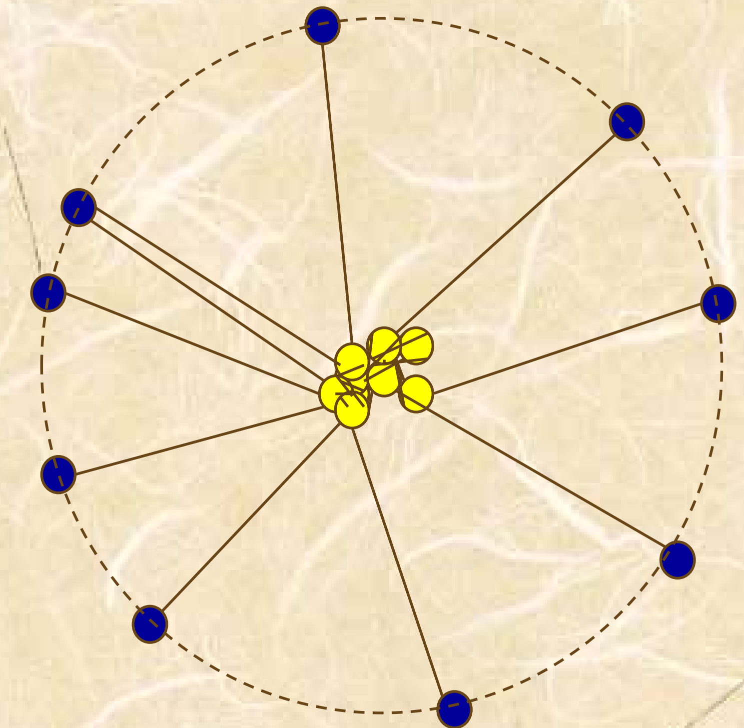


NoGeo (Rao et al., Mobicom'03)



Add remaining nodes

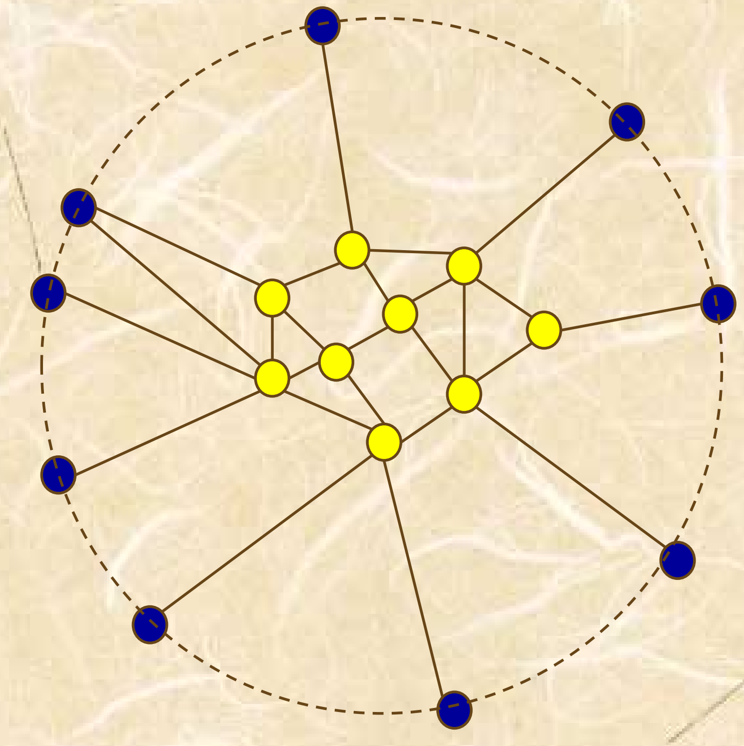
NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Relaxation

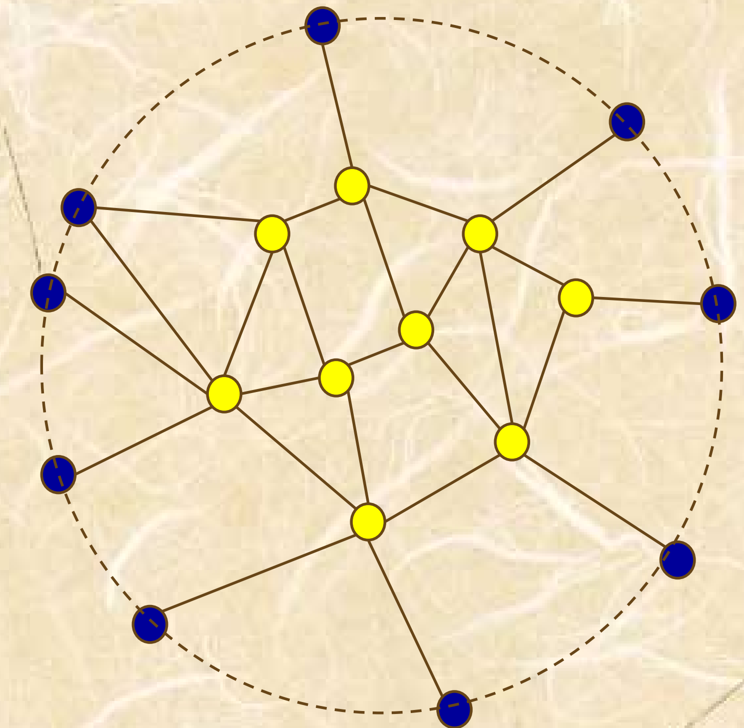
NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Relaxation

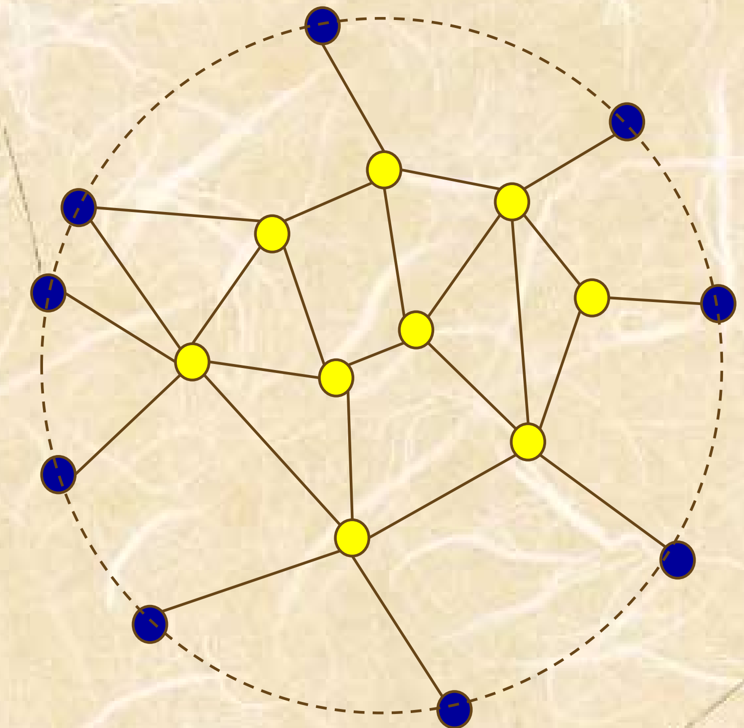
NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Relaxation

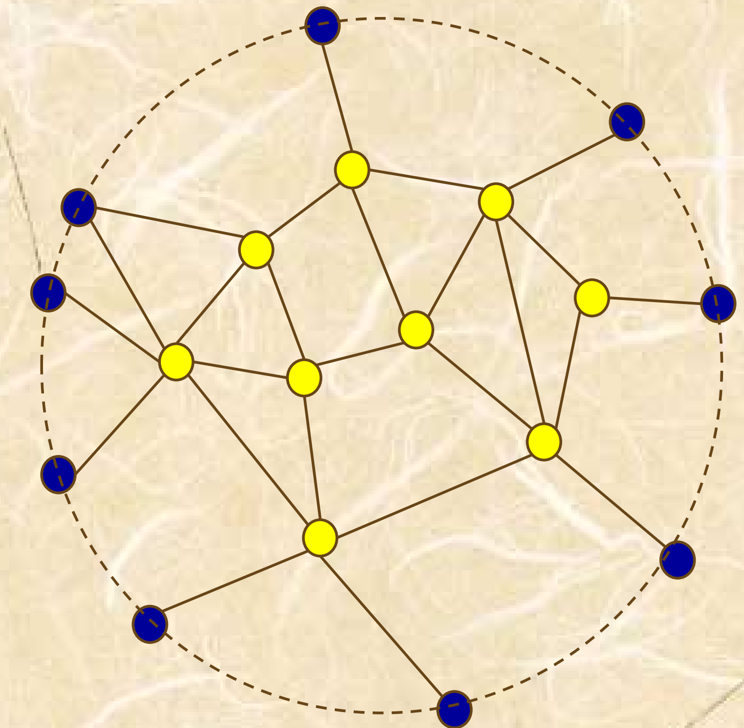
NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Relaxation

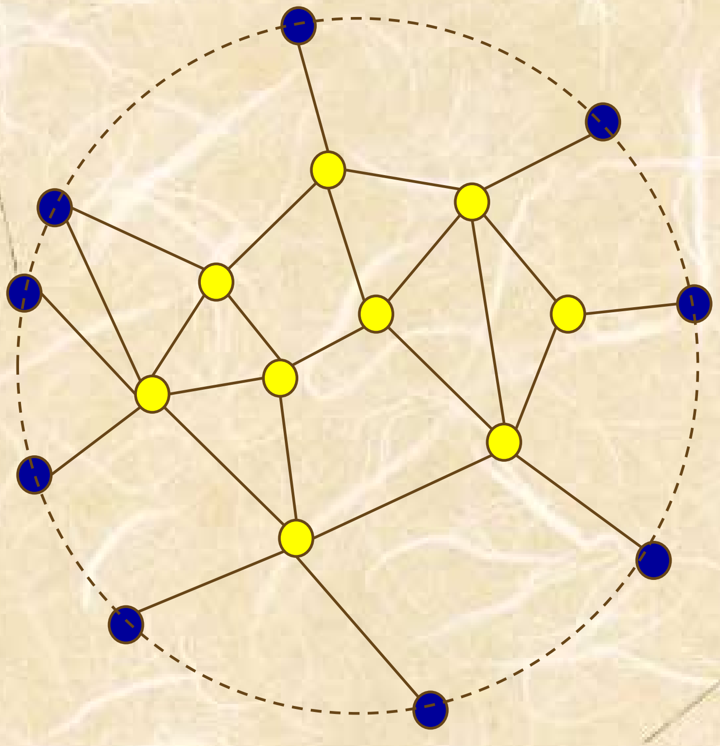
NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Relaxation

NoGeo (Rao et al., Mobicom'03)



$$\vec{x}_i = \frac{1}{n}\sum_k \vec{x}_k,$$ k is a neighbor of i

Convergence!

Issues with NoGeo

- Detect wrong perimeter nodes
- Too many perimeter nodes
  - $O(p^2)$ broadcast messages
- Relaxation is slow
- Relaxation is not "stable"
- Limitation: 2D algorithm (what about 3D networks?)

Summary

- Select reference node
- Elect perimeter nodes
- Initialization of perimeter nodes
  - Error minimization
  - Project onto circle
- Relaxation
  - average of neighbors

Summary

- Select reference node
- Elect perimeter nodes
- Initialization of perimeter nodes
  - Error minimization
  - Project onto circle
- Relaxation

Key Insights

Summary

- Select reference node
- Too many perimeter nodes
- Initialization of perimeter nodes
  - Error minimization
  - Project onto circle
- Relaxation

Key Insights

Summary

- Select reference node
- Elect only 4 reference nodes
- Initialization of reference nodes
  - Error minimization
  - Project onto circle
- Relaxation

Key Insights

Summary

- Select reference node
- Elect only 4 reference nodes
- Initialization of reference nodes
  - Error minimization using geometry
  - Project onto circle
- Relaxation

Key Insights

Summary

- Select reference node
- Elect only 4 reference nodes
- Initialization of reference nodes
  - Error minimization using geometry
  - ~~Project onto circle~~
- Relaxation

Key Insights

Summary

- Select reference node
- Elect only 4 reference nodes
- Initialization of reference nodes
  - Error minimization using geometry
  - ~~Project onto circle~~
- Spring Relaxation

Key Insights

What GSpring got Right

- Select reference node
- Elect only 4 reference nodes ✓
- Initialization of reference nodes
  - Error minimization using geometry ✗
  - ~~Project onto circle~~
- Spring Relaxation

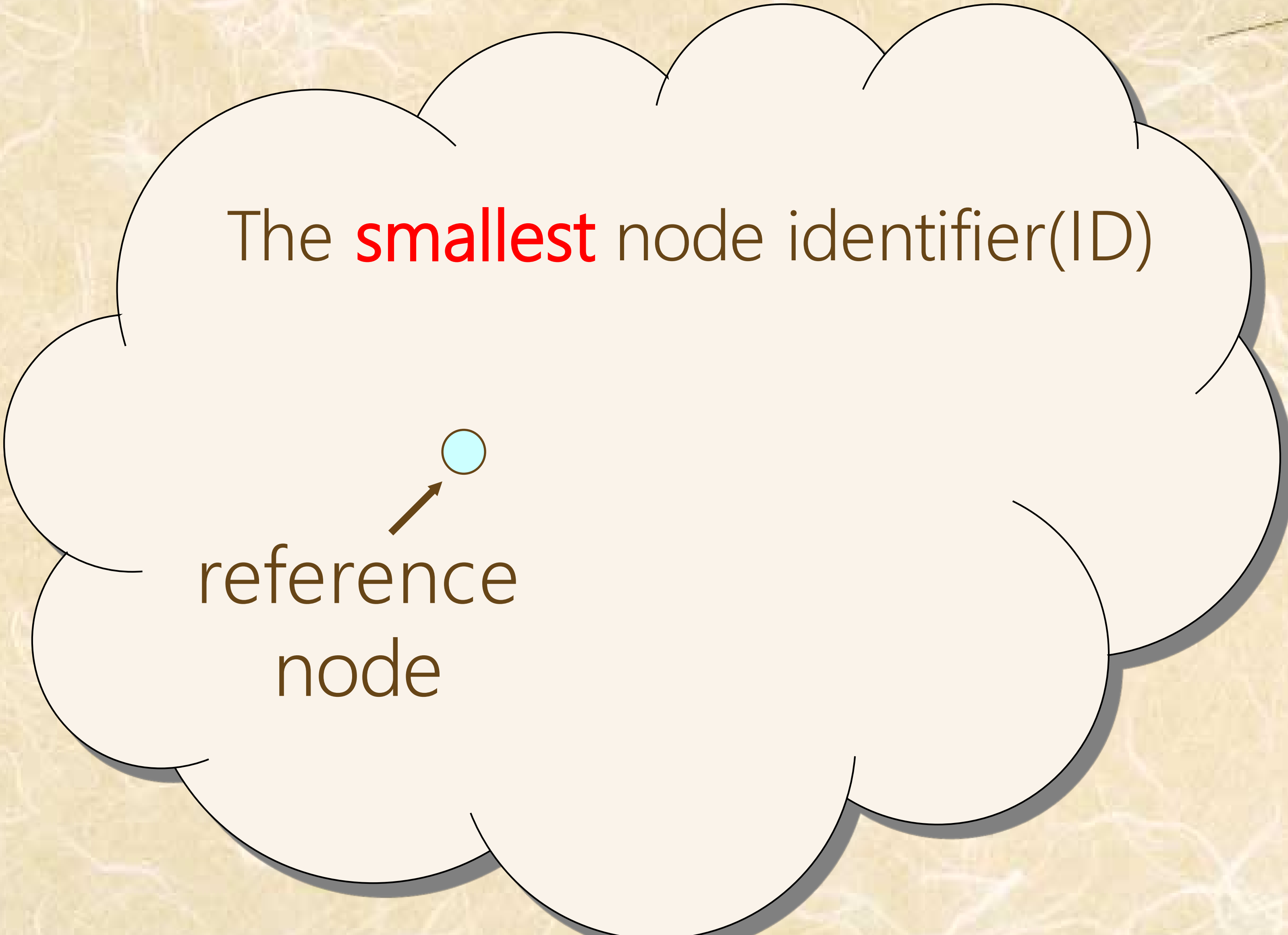
Key Insights

What GSpring got Right

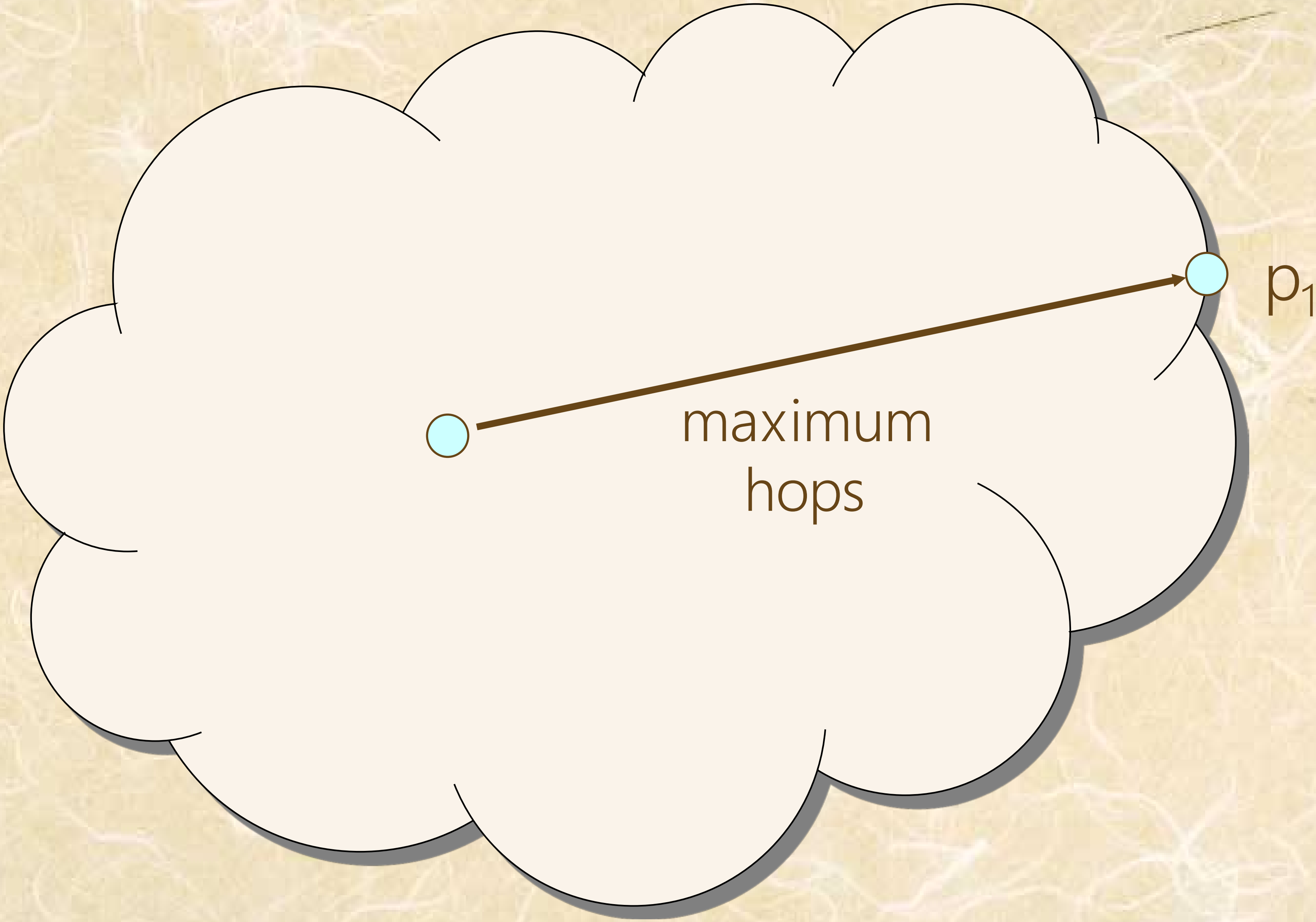
- Select reference node
- Elect only 4 reference nodes ✓
- Initialization of reference nodes
  - Error minimization using geometry ✗
  - Smarter Projection on circle ✗
- Spring Relaxation ✓ **+ Extra** ✗

Key Insights

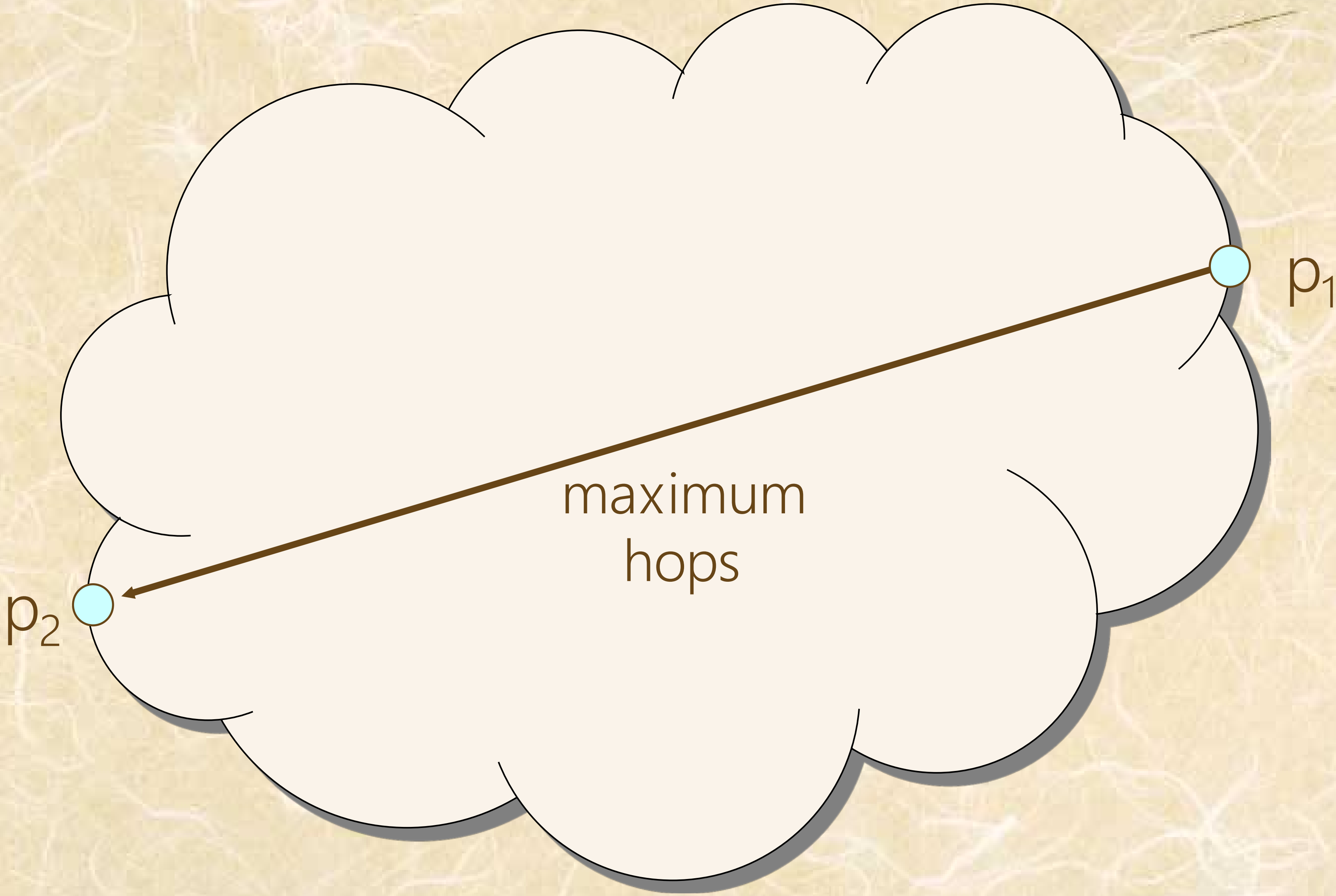
Particle Swarm Virtual Coordinates



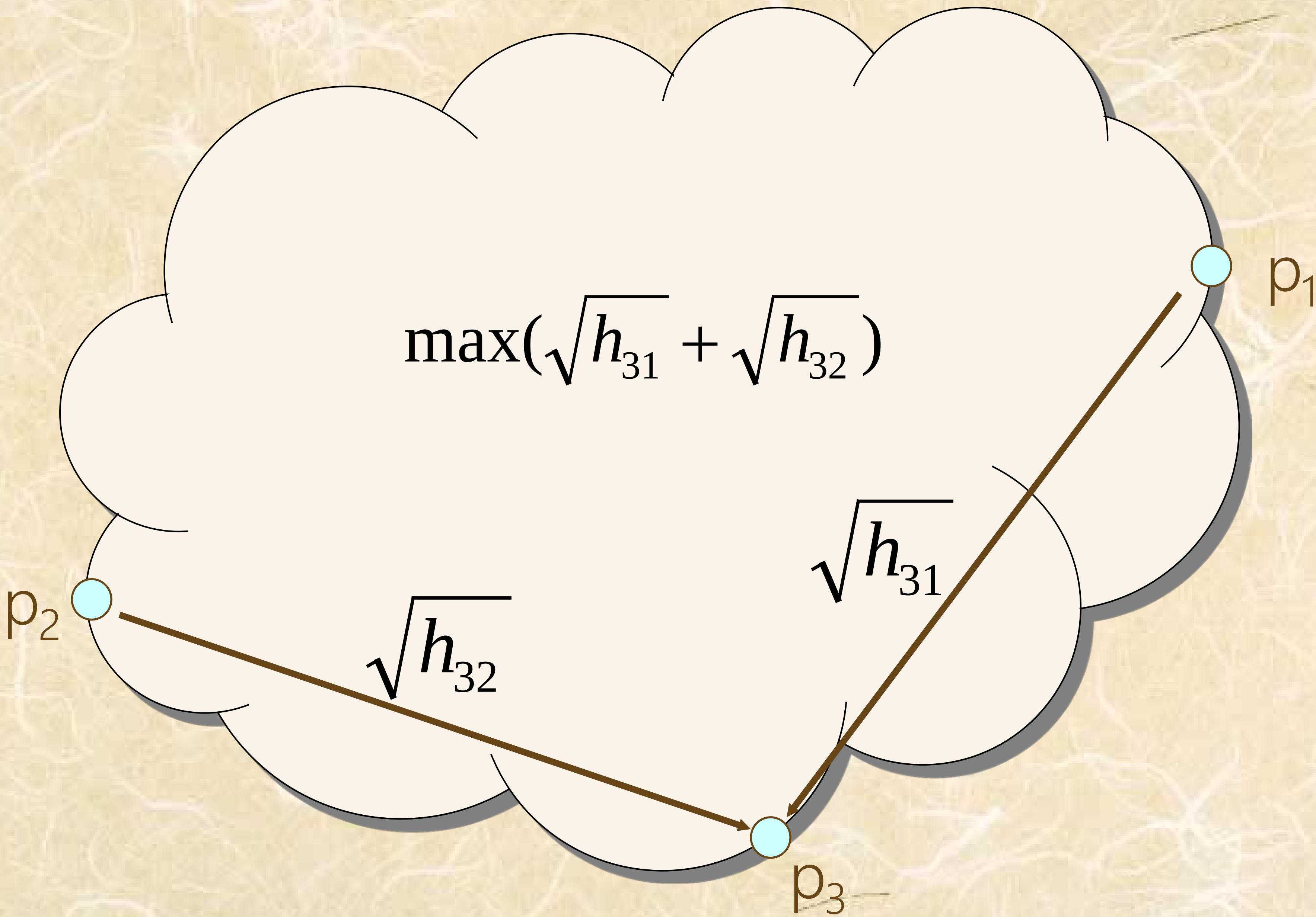
Particle Swarm Virtual Coordinates



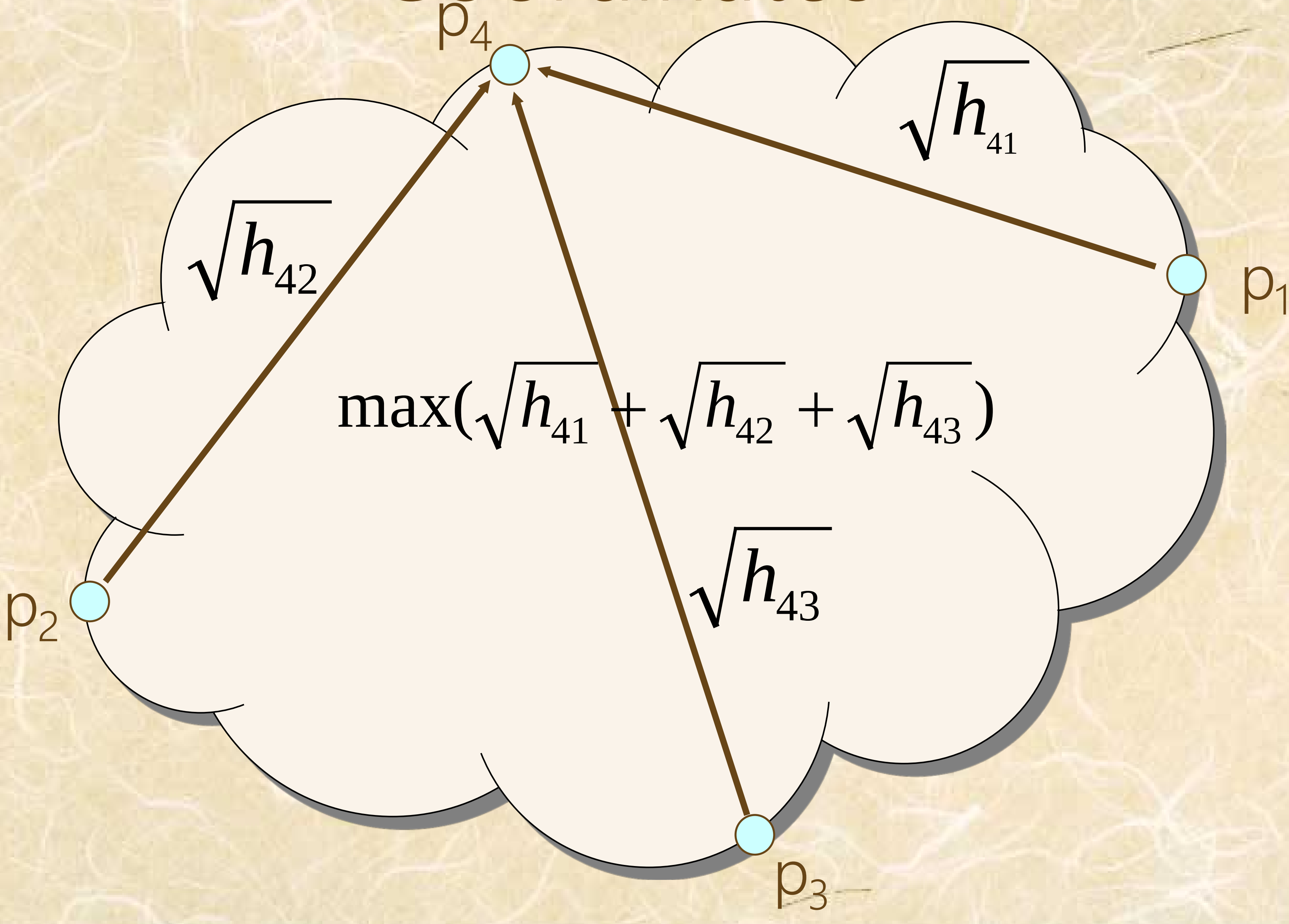
Particle Swarm Virtual Coordinates



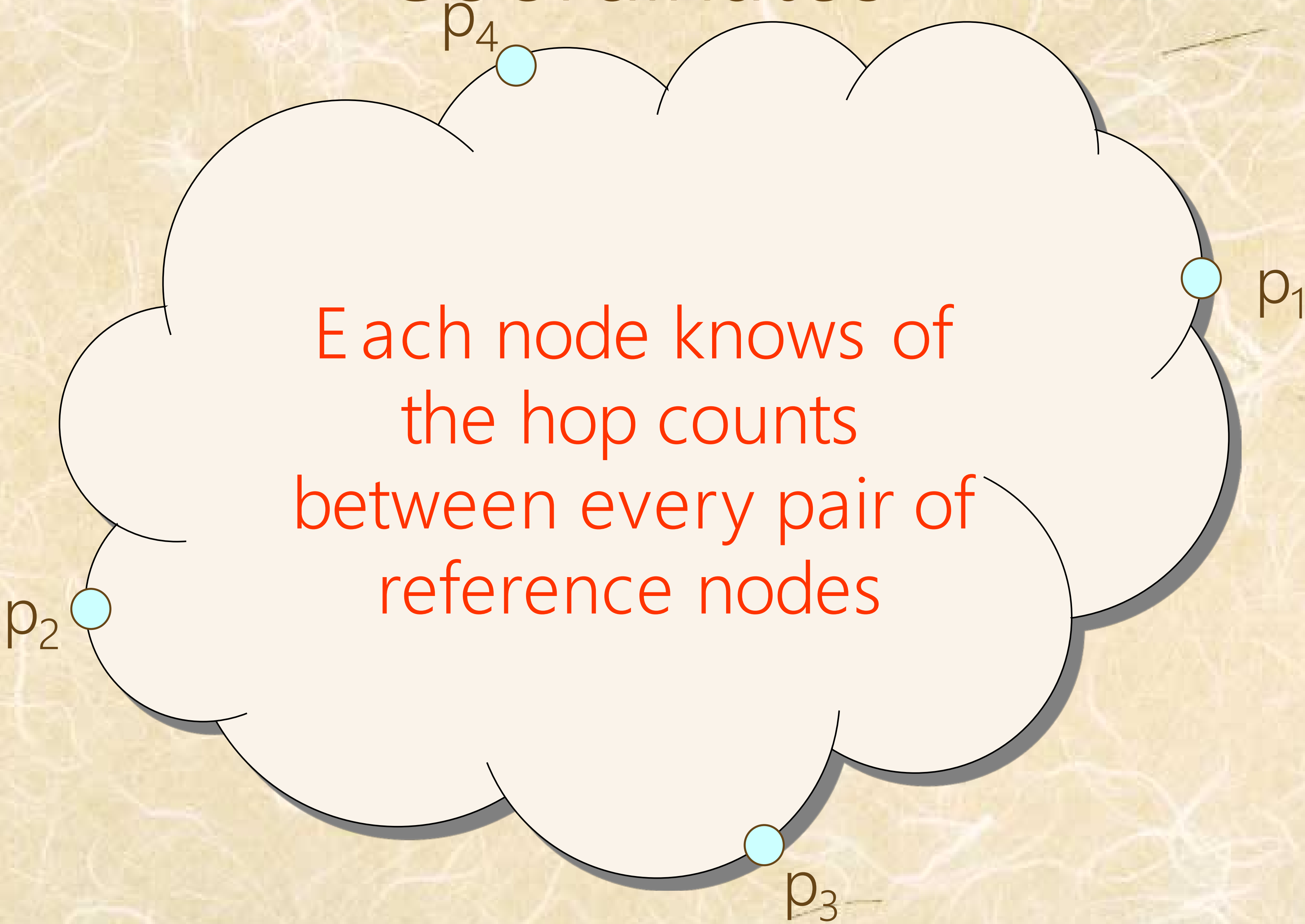
Particle Swarm Virtual Coordinates



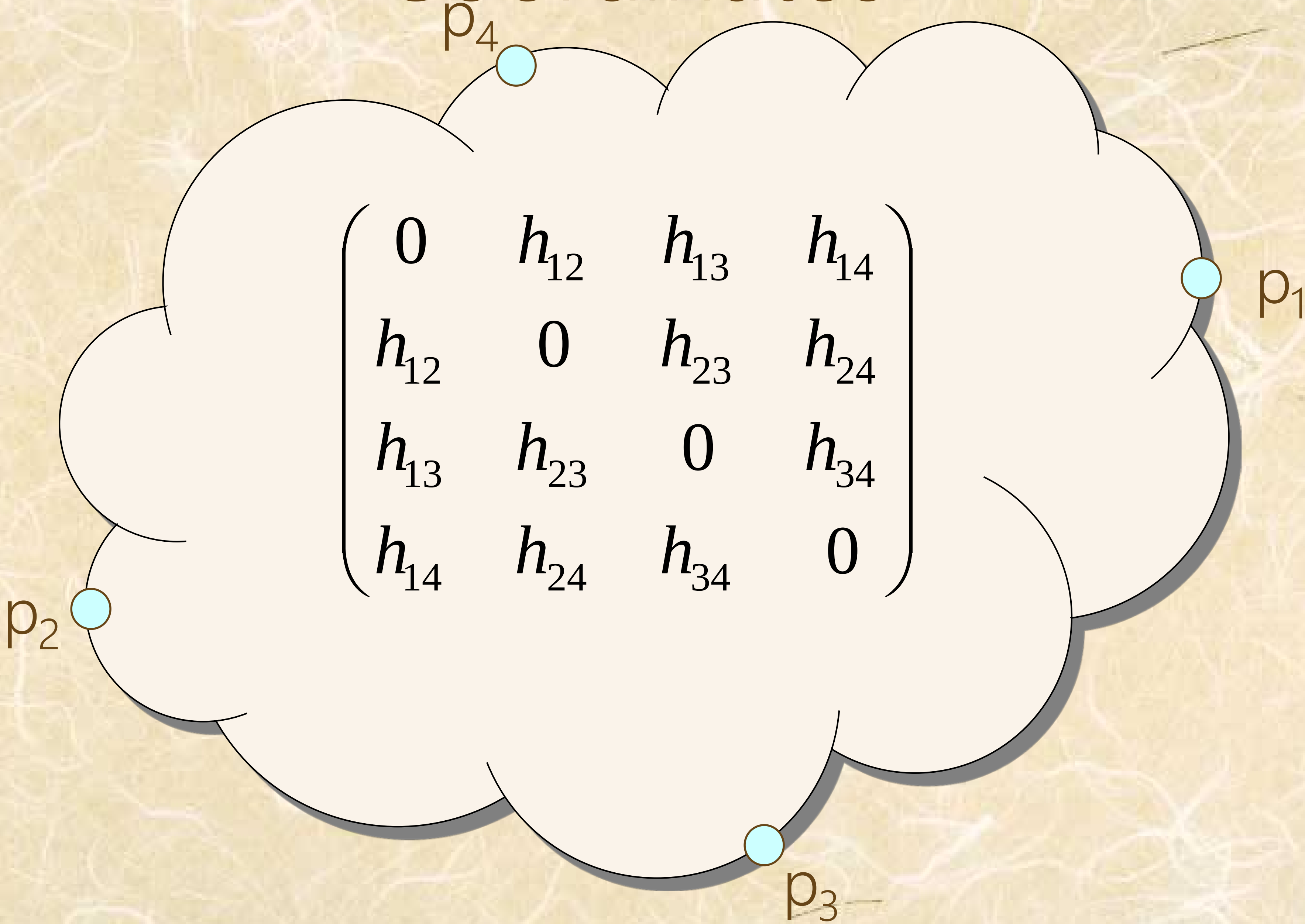
Particle Swarm Virtual Coordinates



Particle Swarm Virtual Coordinates



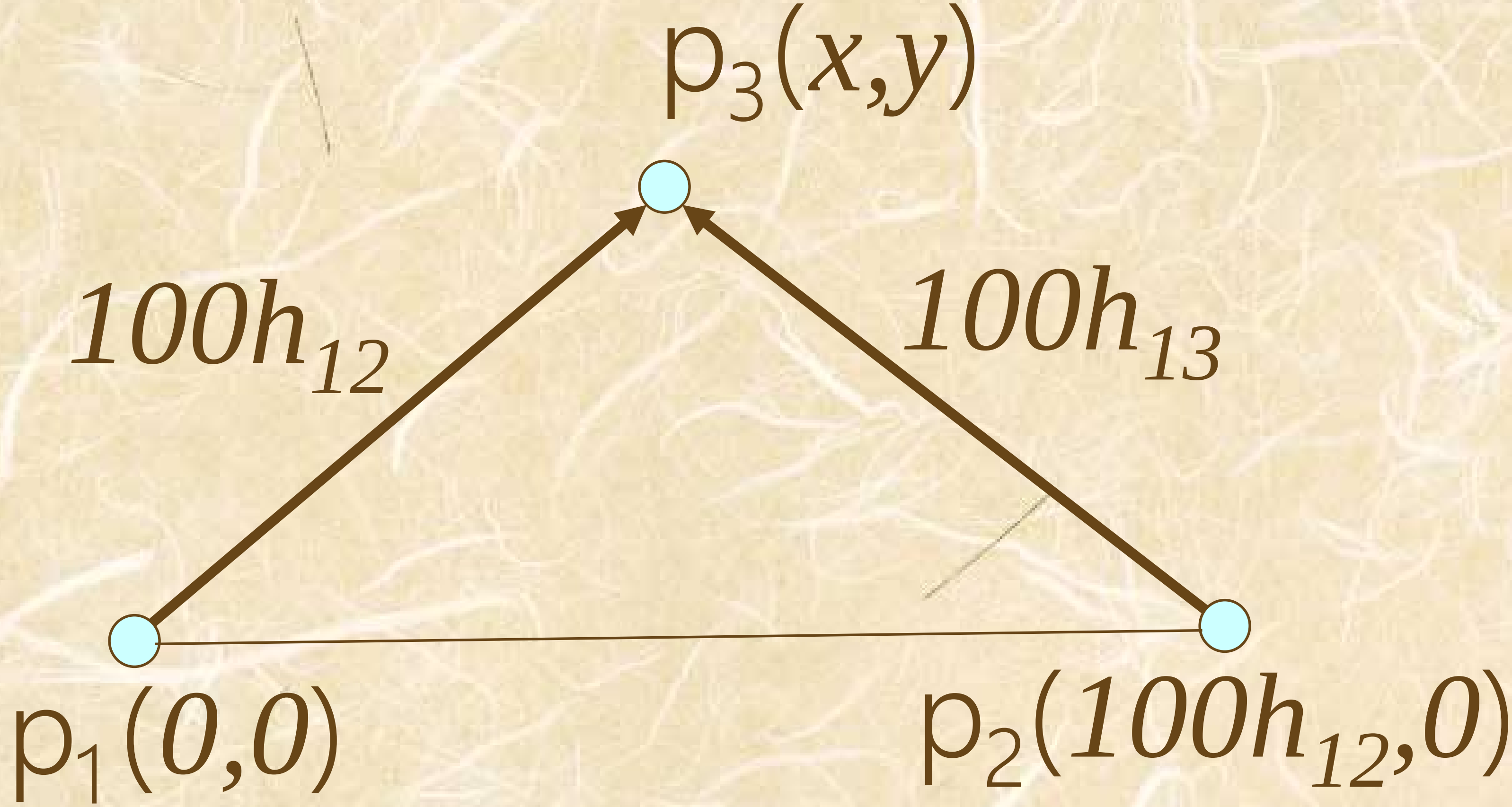
Particle Swarm Virtual Coordinates



$$\begin{pmatrix} 0 & h_{12} & h_{13} & h_{14} \\ h_{12} & 0 & h_{23} & h_{24} \\ h_{13} & h_{23} & 0 & h_{34} \\ h_{14} & h_{24} & h_{34} & 0 \end{pmatrix}$$

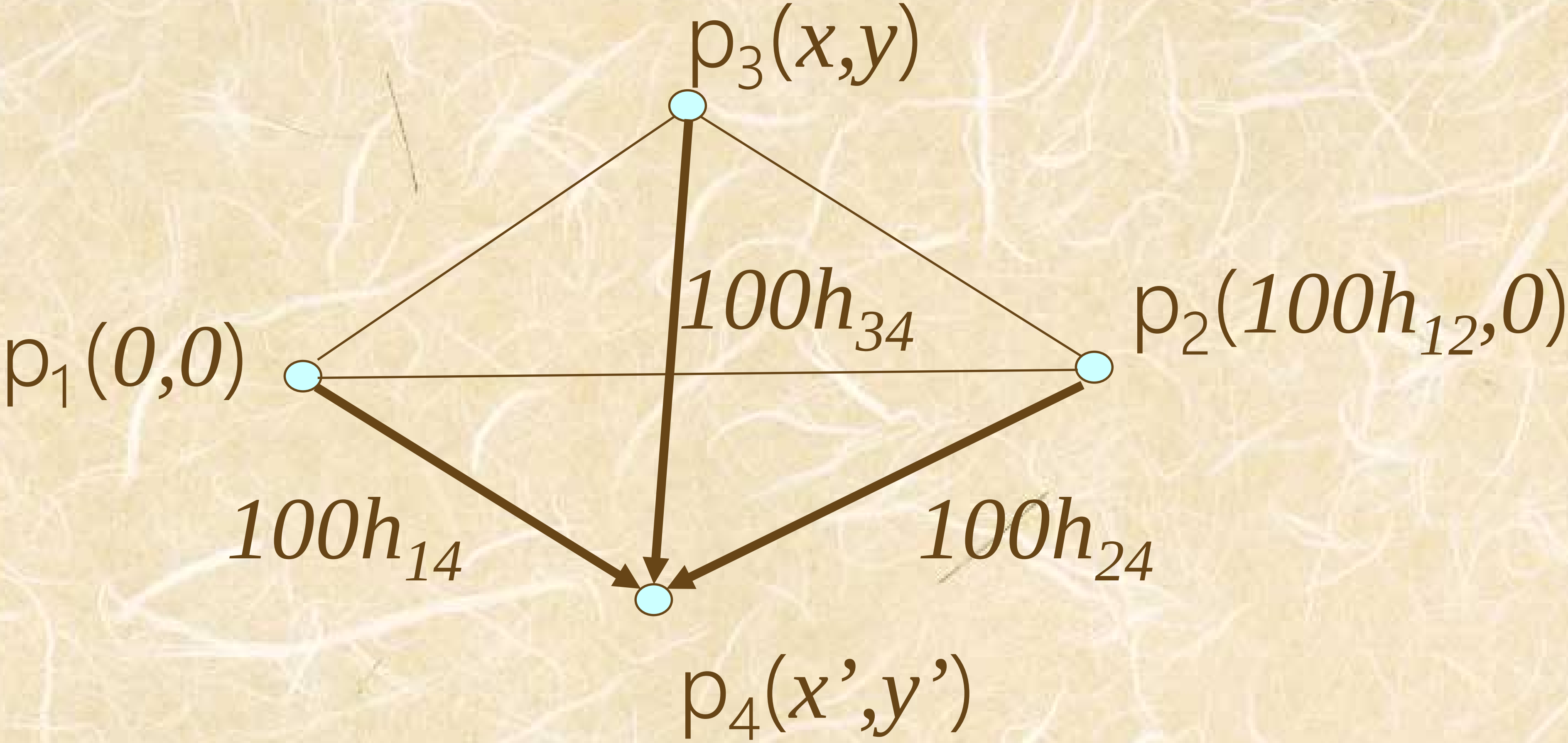
Particle Swarm Virtual Coordinates

Particle Swarm Virtual Coordinates



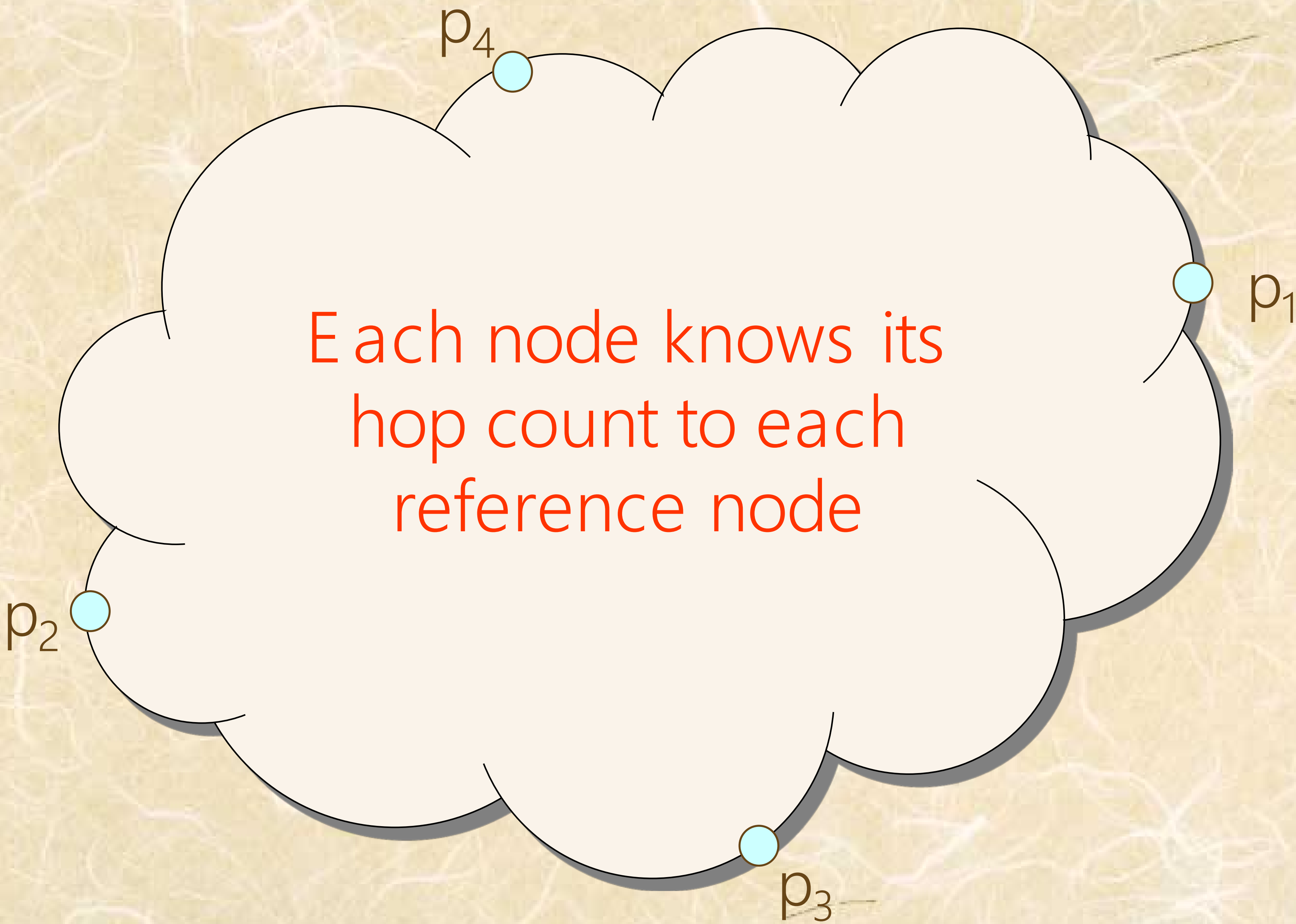
$$\min E = \sum_{i=1}^{k-1} (|\vec{x}_i - \vec{x}_j| - 100h_{ik})^2$$

Particle Swarm Virtual Coordinates

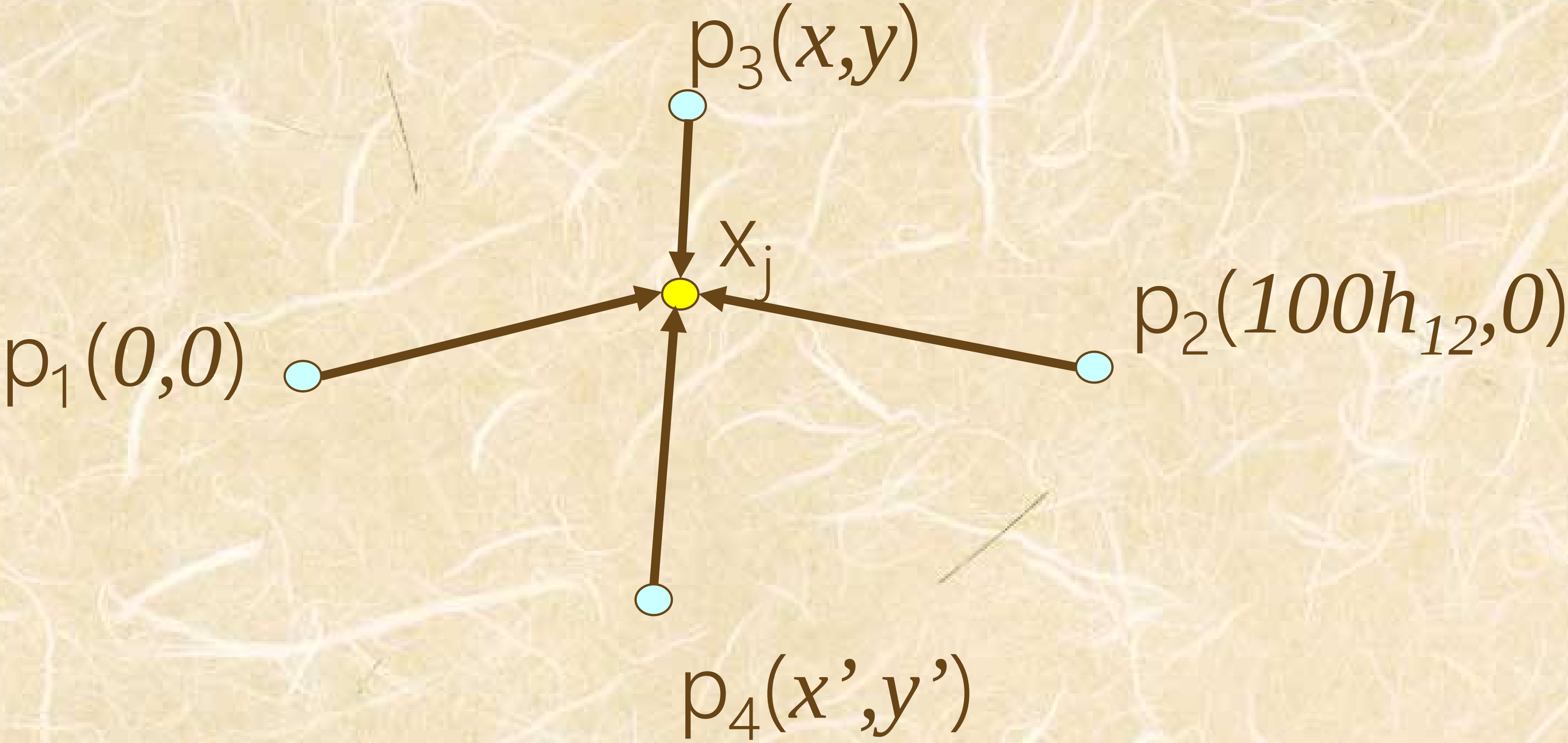


$$\min E = \sum_{i=1}^{k-1} (|\vec{x}_i - \vec{x}_j| - 100h_{ik})^2$$

Observation

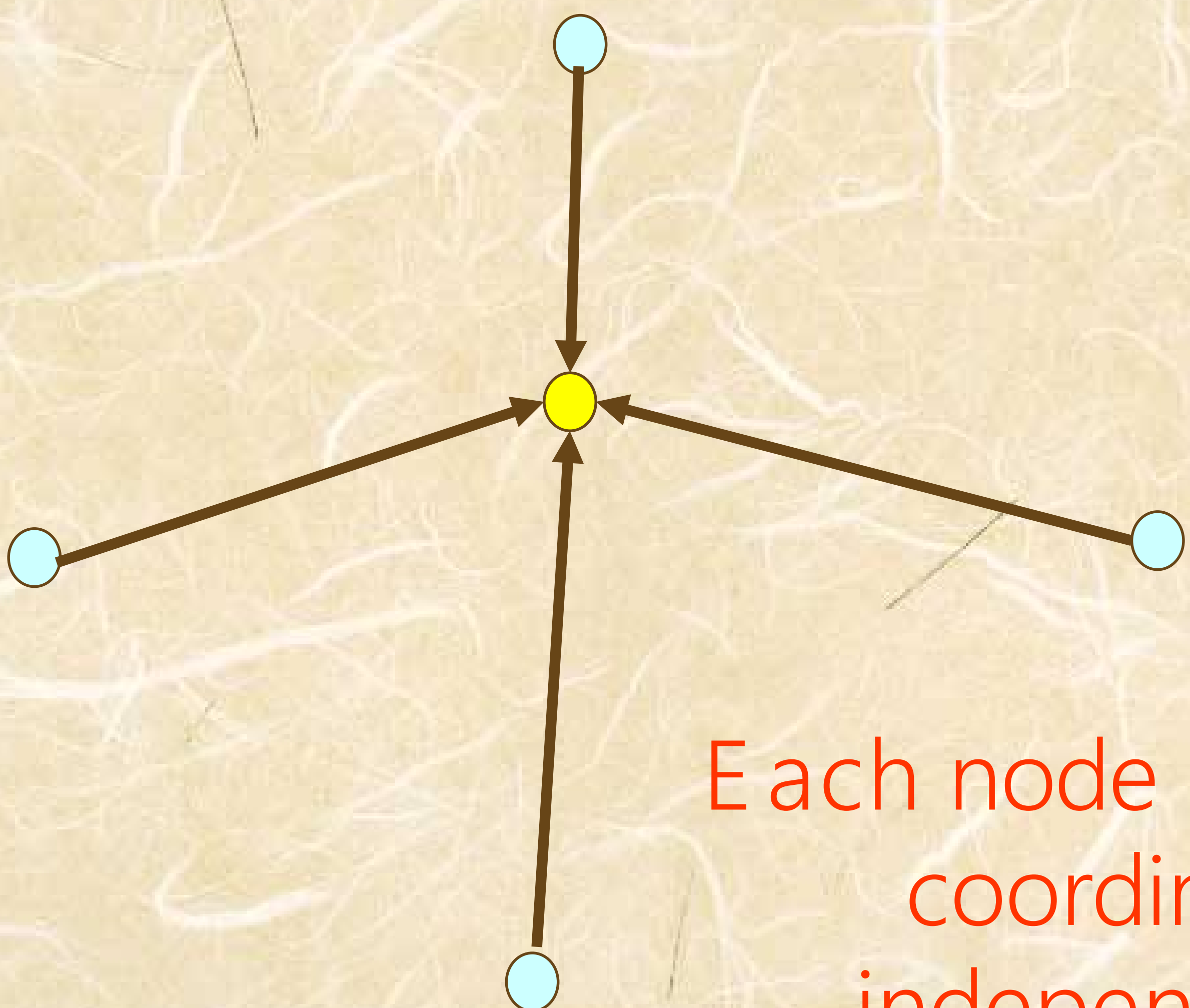


Particle Swarm Virtual Coordinates



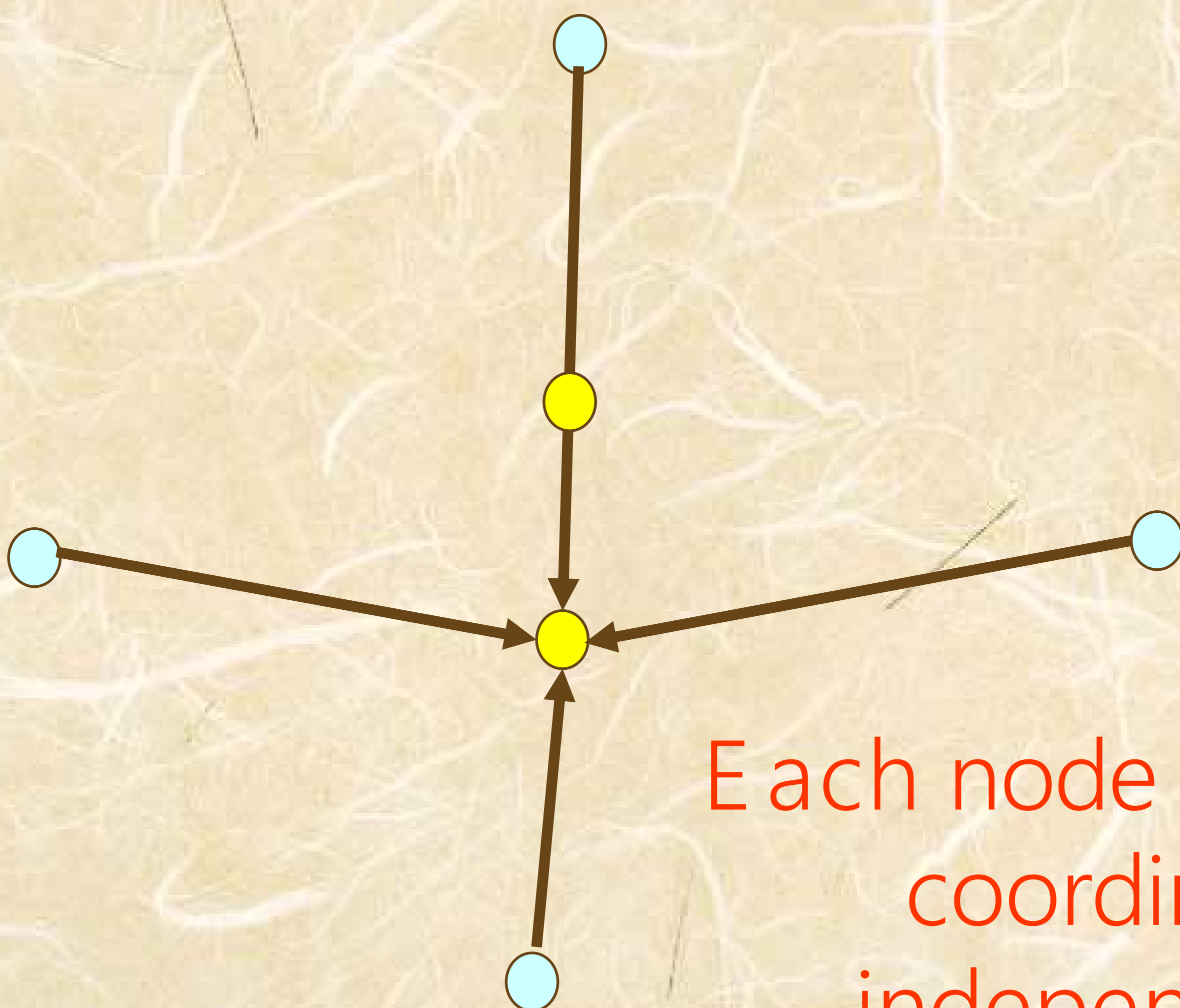
$$\min E_j = \sum_{i=1}^{4} (|\vec{p}_i - \vec{x}_j| - 100h_{ij})^2$$

Particle Swarm Virtual Coordinates



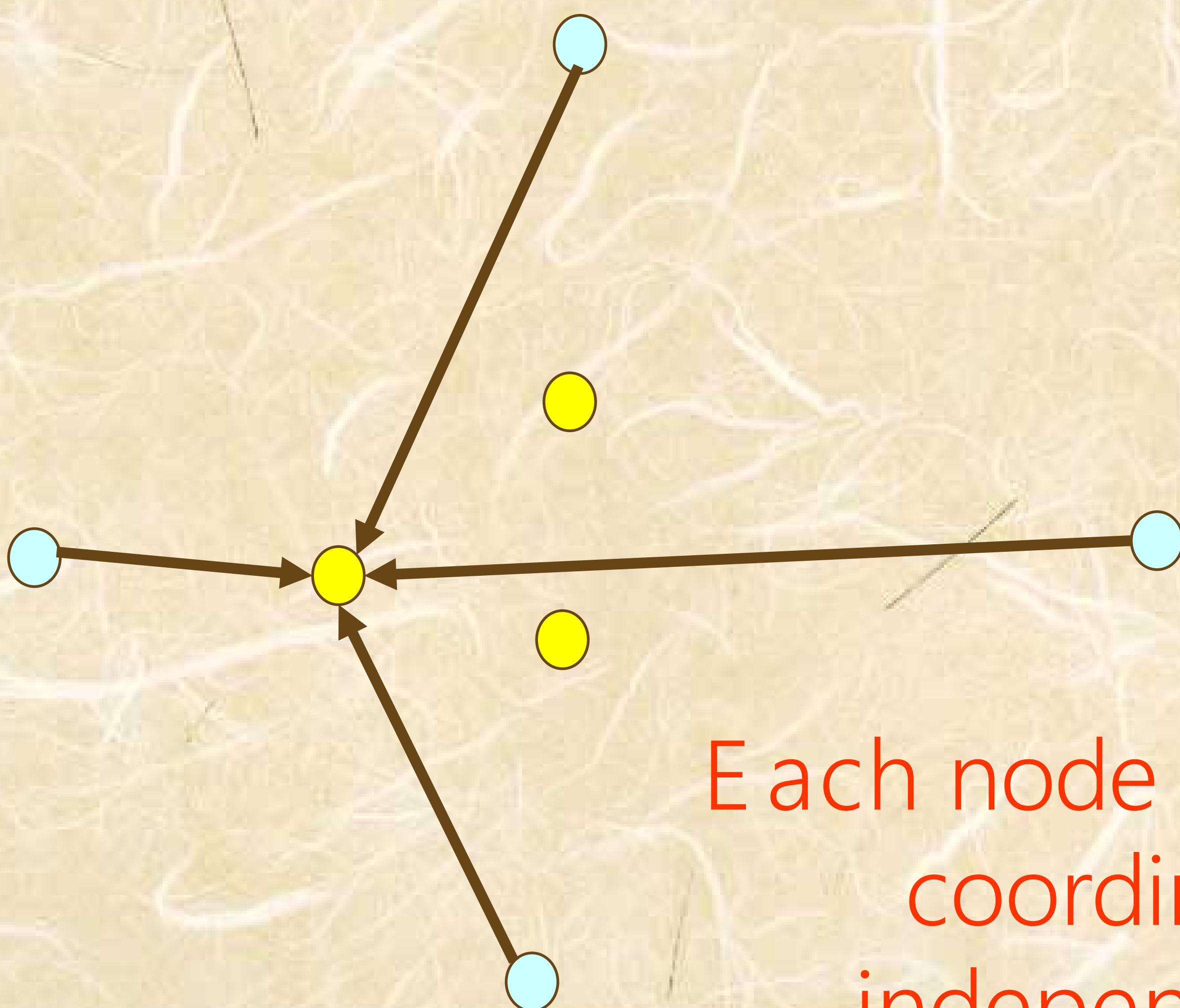
Each node computes coordinates independently

Particle Swarm Virtual Coordinates



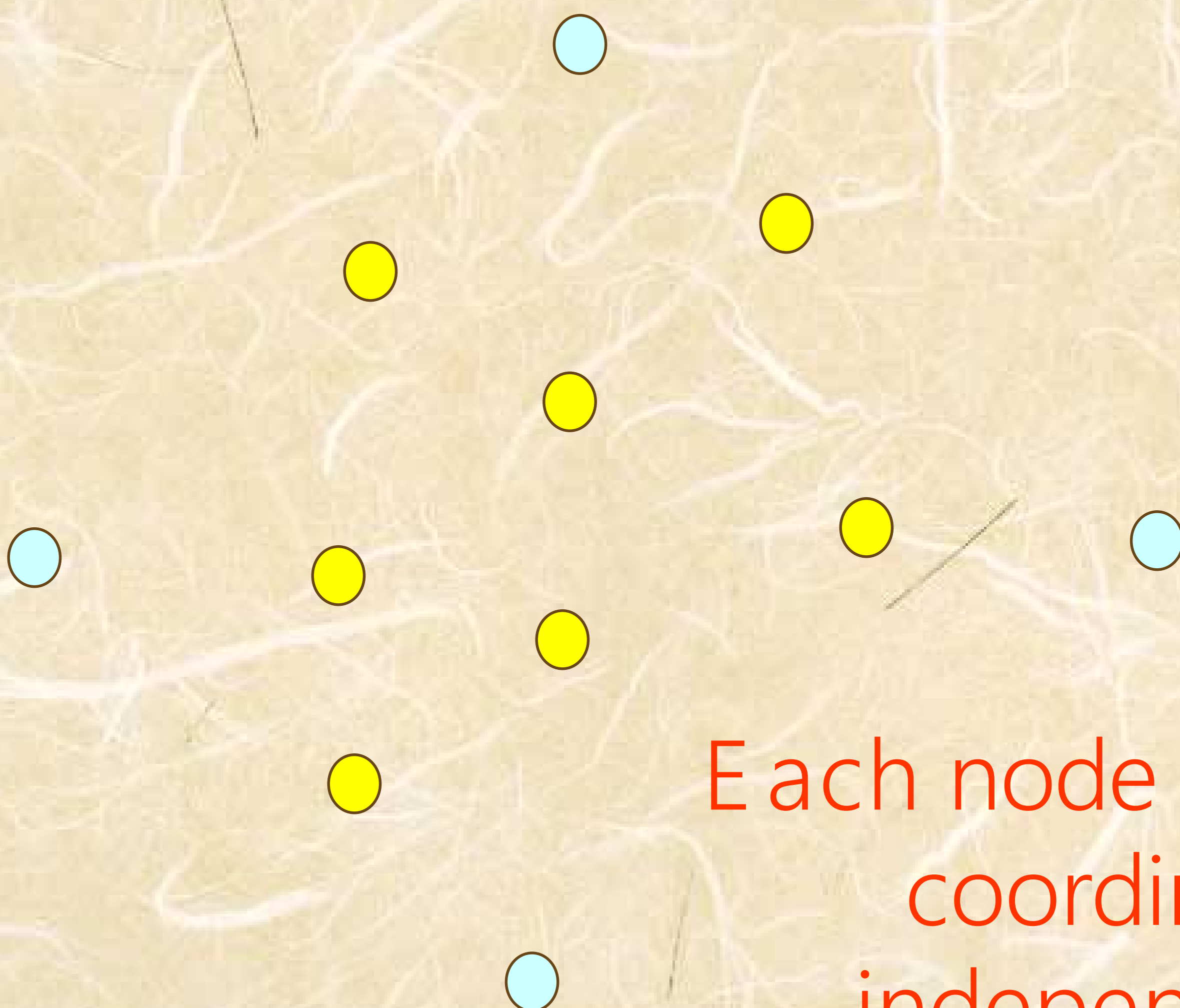
Each node computes coordinates independently

Particle Swarm Virtual Coordinates



Each node computes coordinates independently

Particle Swarm Virtual Coordinates



Each node computes coordinates independently

Particle Swarm Virtual Coordinates

Spring Relaxation

Spring Relaxation

- Spring force:

$$\overrightarrow{F_{ij}} = \kappa \times (l_{ij} - |\overrightarrow{x_i} - \overrightarrow{x_j}|) \times u(\overrightarrow{x_i} - \overrightarrow{x_j})$$

(Hooke's Law)

- Net force:

$$\overrightarrow{F_i} = \sum_{j \neq i} \overrightarrow{F_{ij}}$$

THAT'S PSVC!!

- Update rule:

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \frac{\min(|\overrightarrow{F_i}|, \alpha_t)}{|\overrightarrow{F_i}|} \overrightarrow{F_i}$$

Wait A Moment… .

Where's the Particle Swarm?

Recall

$$\min E = \sum_{i=1}^{k-1} (|\vec{x}_i - \vec{x}_j| - 100 h_{ik})^2$$

$$\min E_j = \sum_{i=1}^{4} (|\vec{p}_i - \vec{x}_j| - 100 h_{ij})^2$$

Particle Swarm Optimization:

beam & local hill-climbing search (details in paper)

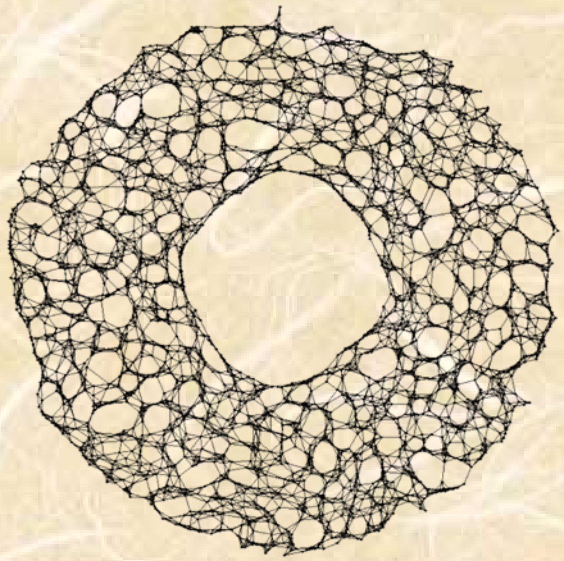
Key ideas

1. Need only a small number of reference nodes
   ⇒ need to choose them well
2. Hop count good proxy for virtual distance
3. Spring relaxation improves convexity
   ⇒ increase greedy forwarding success rates

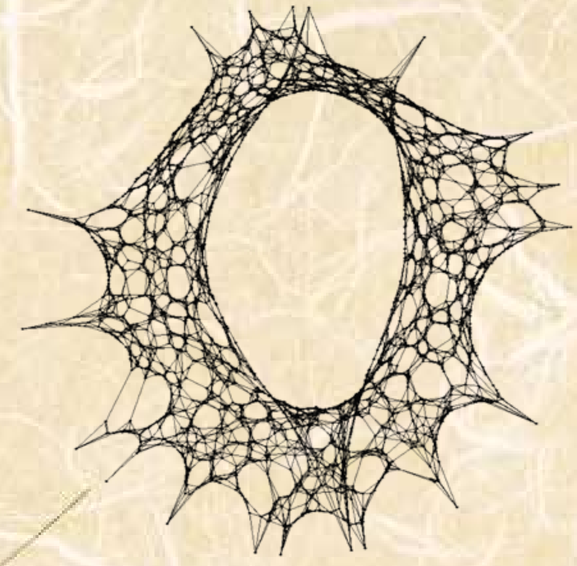
Perfomance Evaluation (TOSSIM)

3,200-node Donut Network

Actual

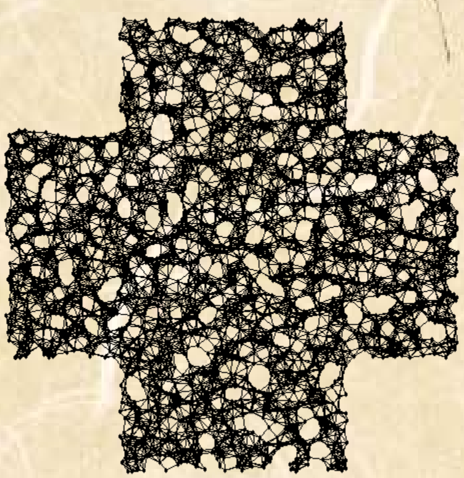


PSVC

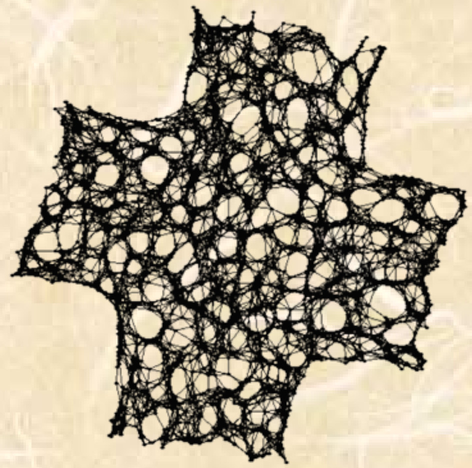


NoGeo

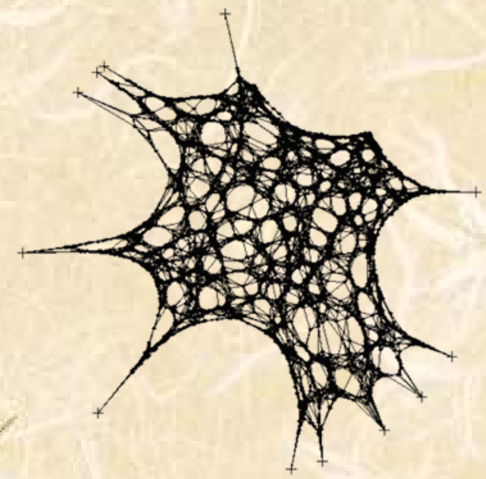
3,200-node Cross Network



Actual

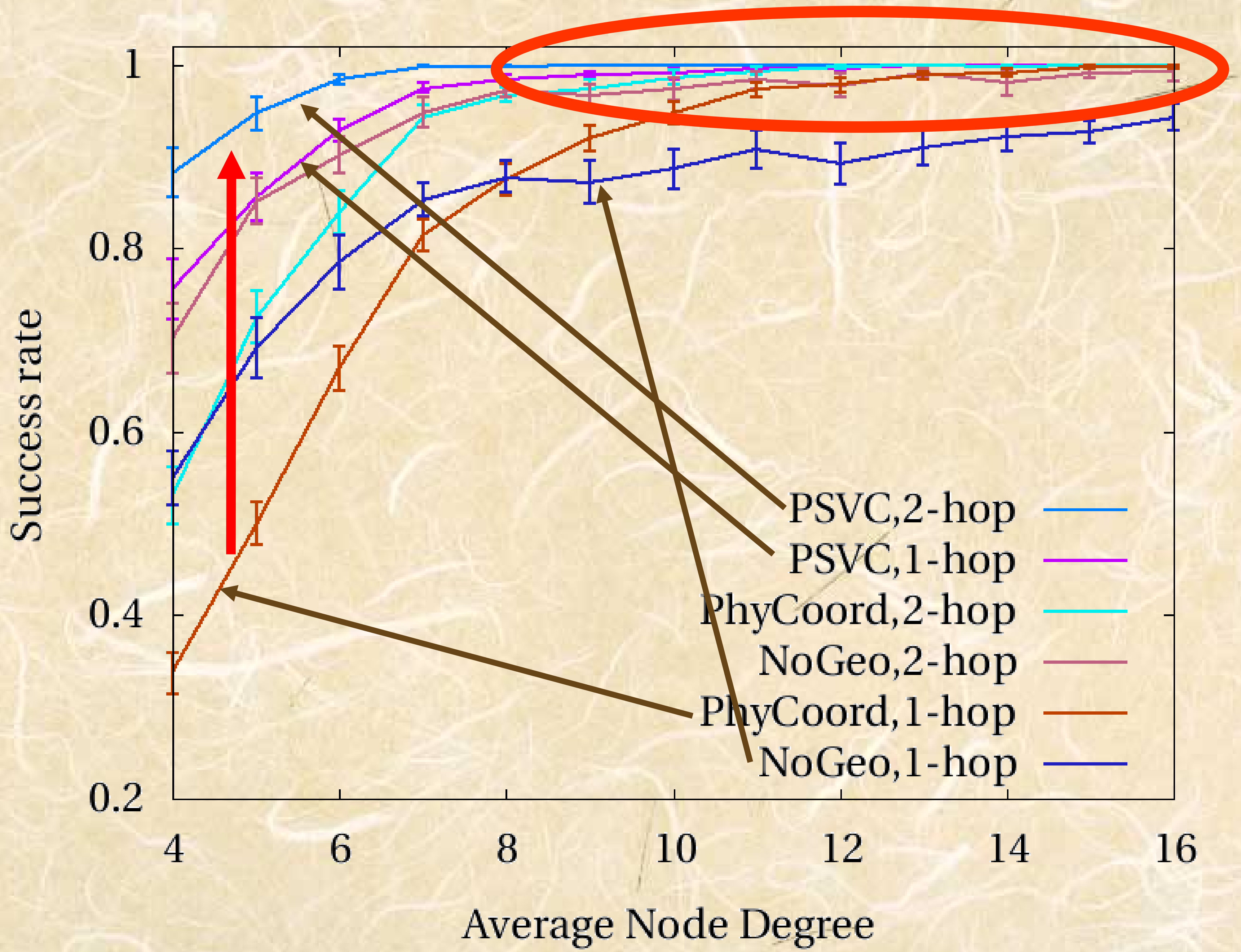


PSVC



NoGeo

Greedy forwarding success rate vs. network density

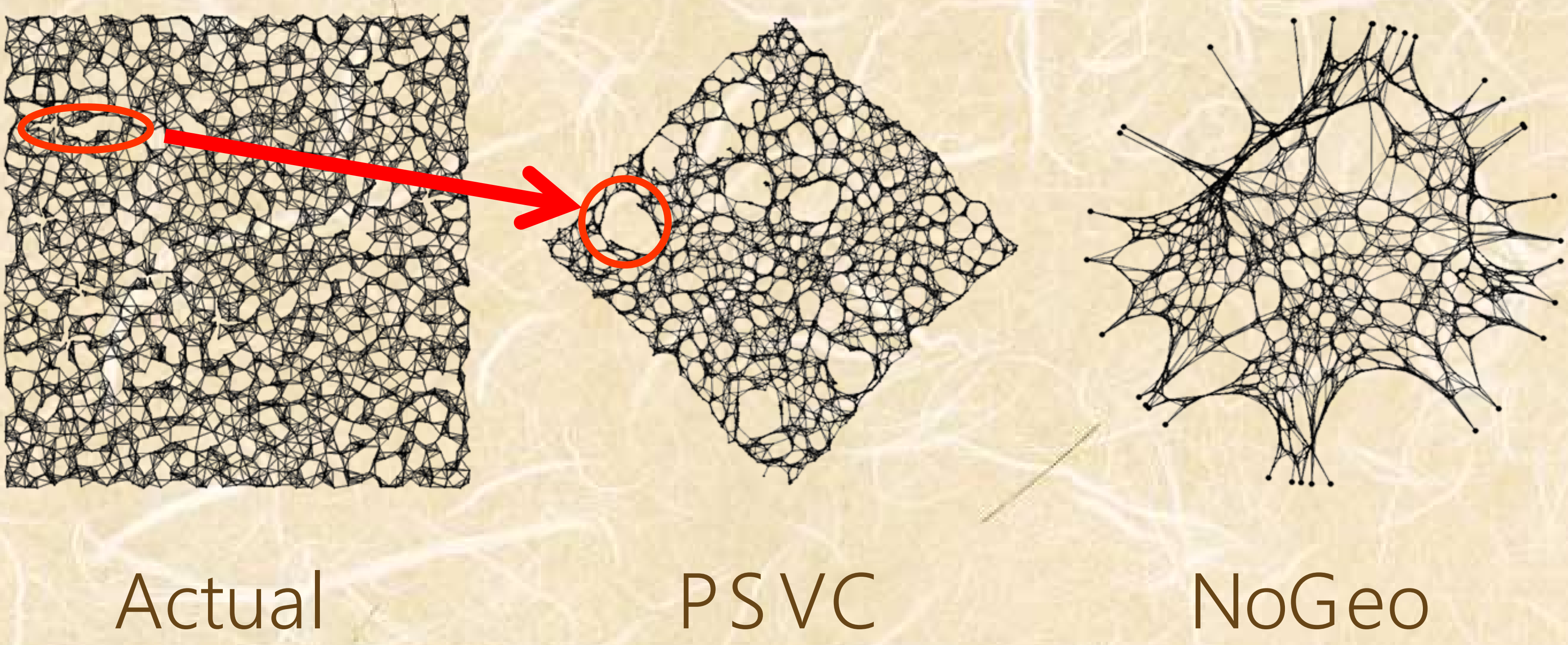


Performance: Hop Stretch (GDSTR) (3,200-node networks)

Network Type	Physical coordinates	NoGeo
Sparse UDG (degree 10)	3% lower	10% lower
Dense UDG (degree 16)	Same	4% lower
Obstacle (degree 8)	35% lower	35% lower

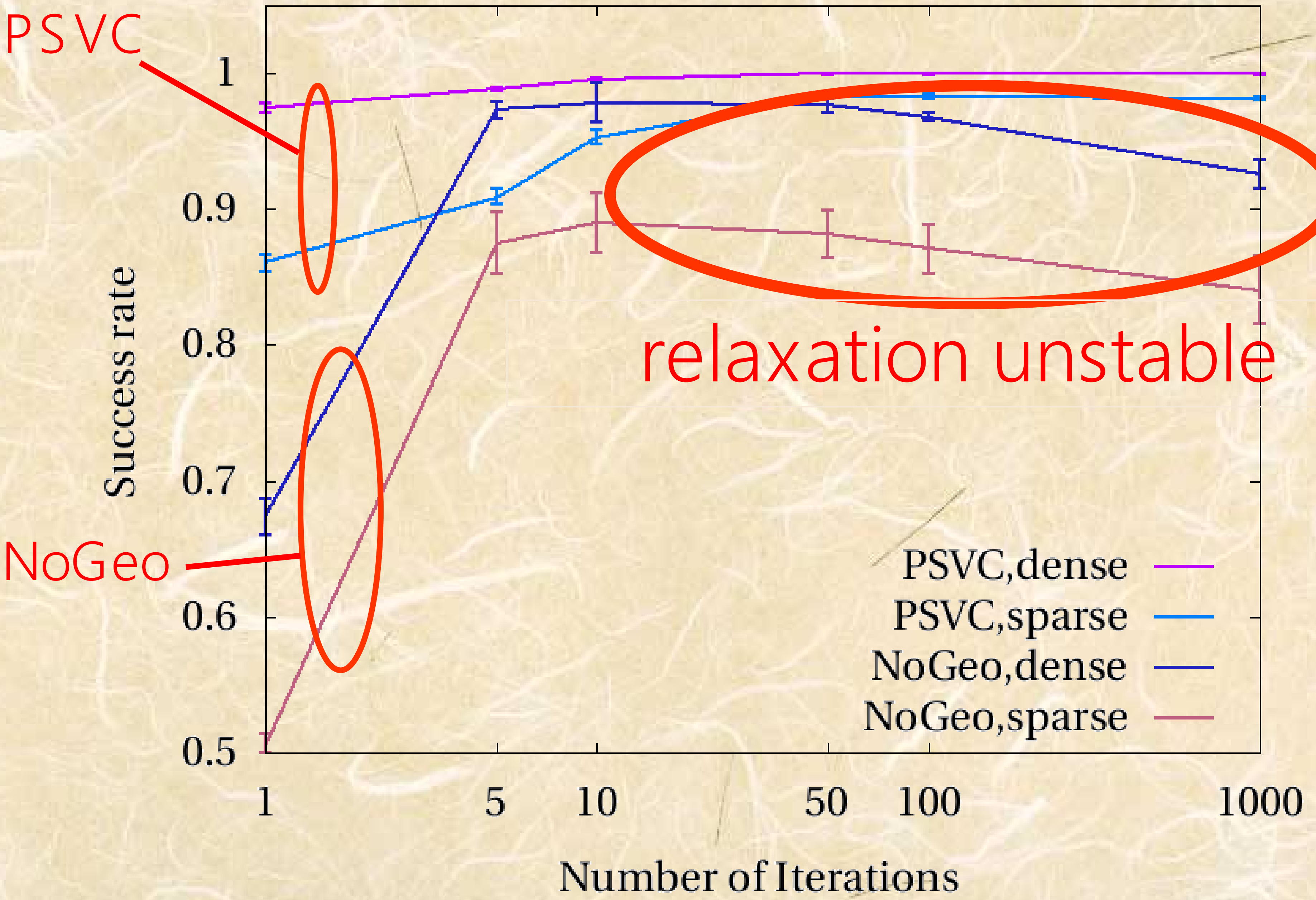
Can do better than actual coordinates!

3,200-node Network with Obstacles

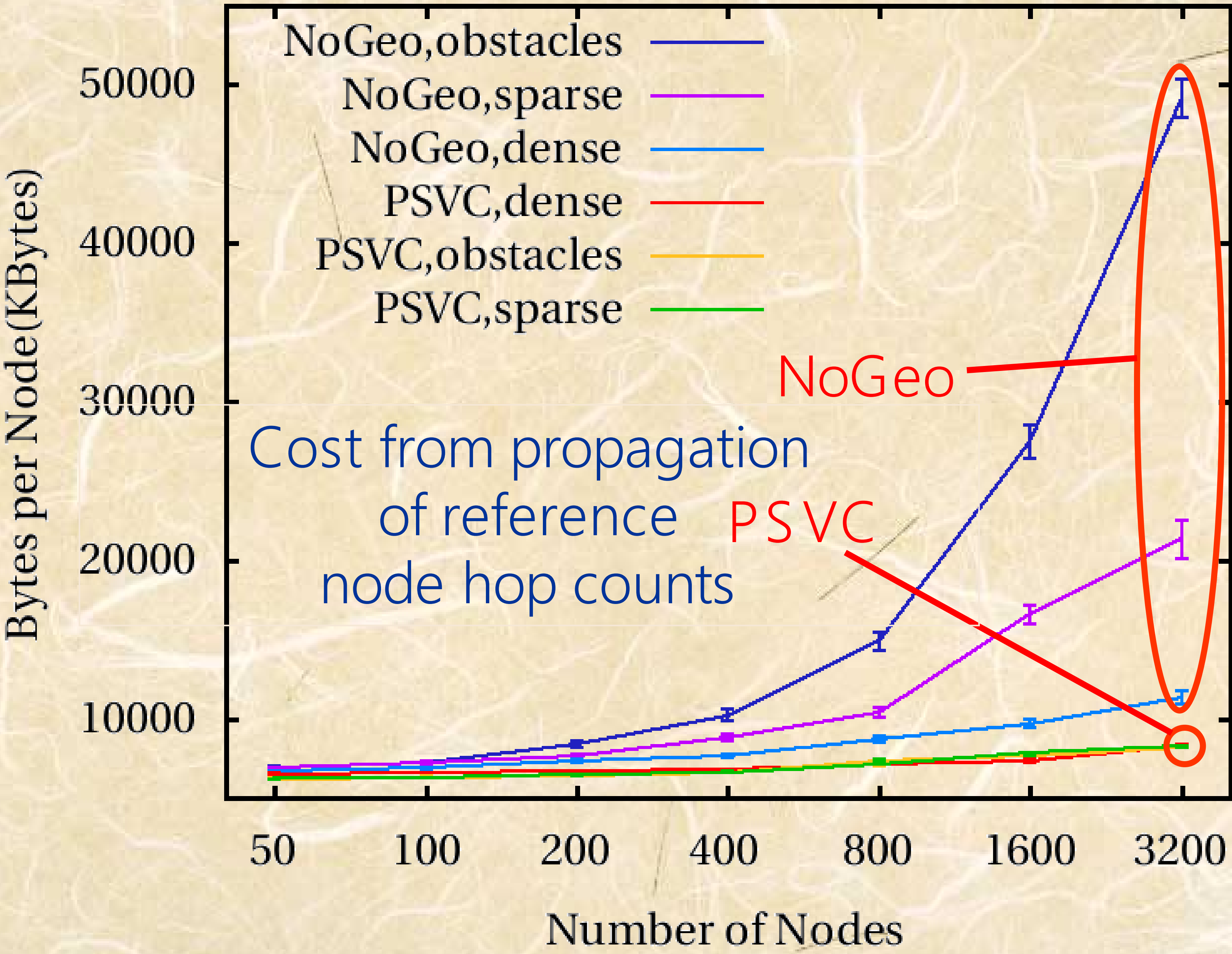


Coordinate assignment + Spring relaxation
⇒ increase convexity

Greedy forwarding success rate vs. Iterations (for 3,200-node 2D networks)



Overhead per node vs. Network size



Limitations/Future Work

- PSVC fits in 48 KB TelosB executable memory
- NoGeo, PSVC/GDSTR cannot fit
- Improve memory footprint
- Evaluation for incremental growth and node failures

More details in paper

- PSO Algorithm
- Details of PSVC
- PSVC easily extensible to 3D
- Comparison of storage costs
- Two-hop greedy forwarding can improve performance significantly
- Evaluations on 120-node Indriya TelosB testbed

TinyOS Source Code

Available here:

https://sites.google.com/site/geographicrouting

Conclusion

- Routing stretch
  - Lower than NoGeo
  - Comparable to actual physical coordinates
  - Superior for networks with obstacles
- Converges fast (~10 iterations)
- Works for 3D networks (!)
- Practical : implemented in TinyOS and evaluated in TelosB testbed

QUESTIONS?

THANK YOU

Background

- Geographic routing is a promising approach for wireless networks
  - Achieve close to optimal routing stretch
  - Scale well
  - Routing states is dependent on local network density and not on network size
- Nodes need location information while no location information is useful at hand
  - Employ virtual coordinates

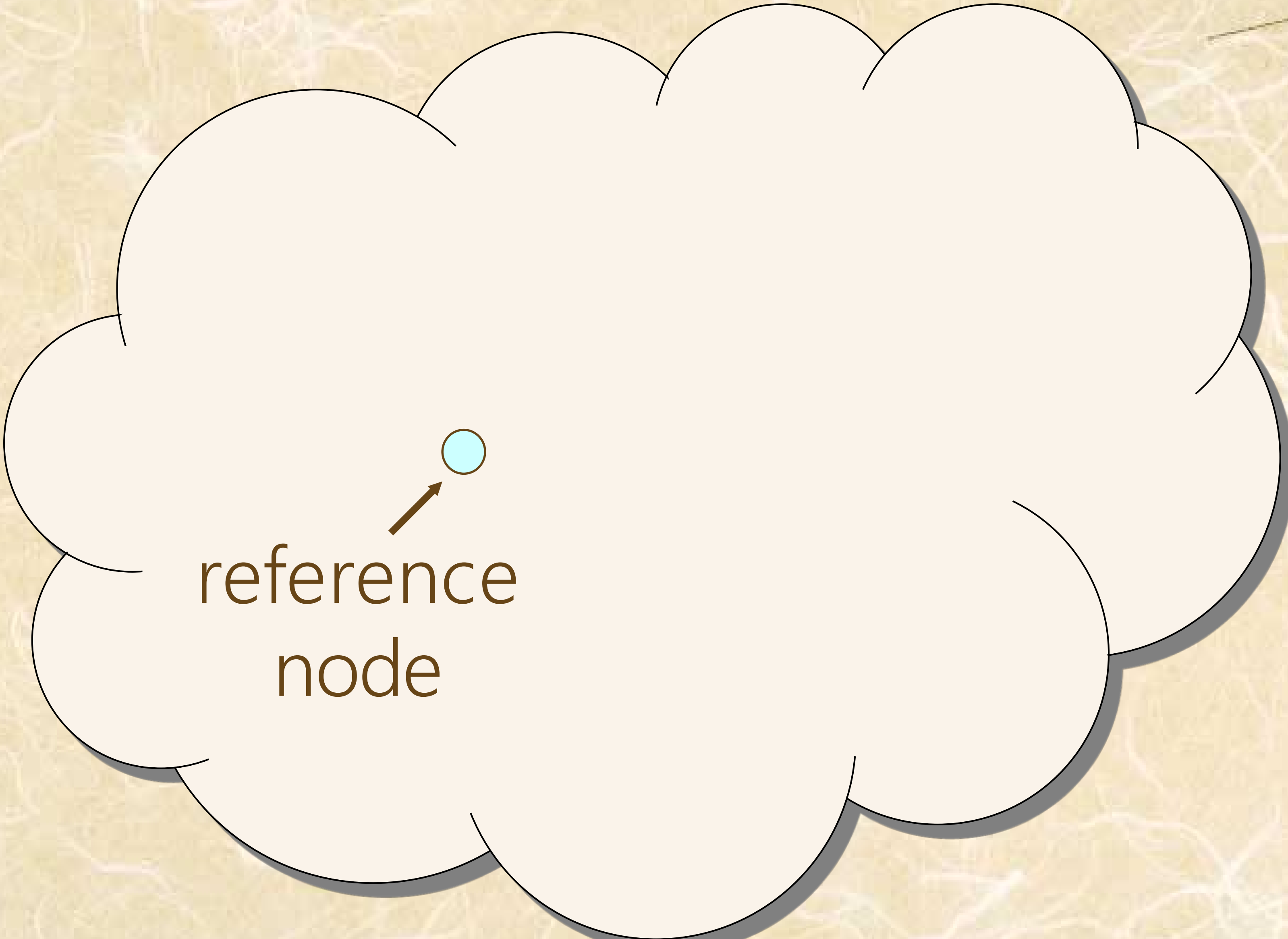
Case for Virtual Coordinates

- Not feasible to manually configure coordinates for each node
- GPS does not work always
- Virtual coordinates are sometimes better, e.g. sensornet on ship
- Actual physical locations are not required (Rao et al., 2003)
- Previous work: good for dense networks and focused on 2D networks
- Know: greedy forwarding is efficient
- Challenge: *can we assign coordinates so that greedy forwarding always works even for 3D networks?*

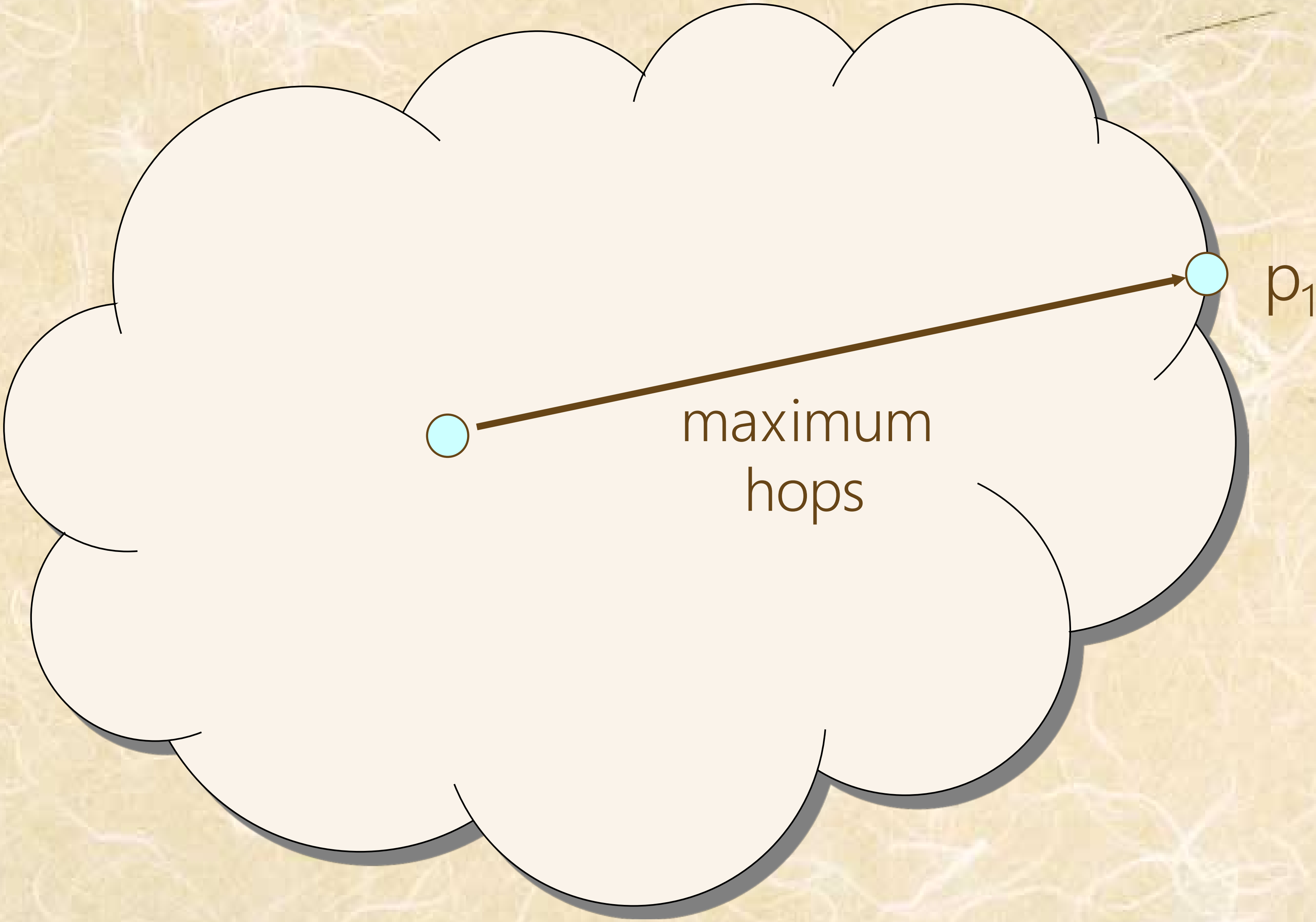
Related Work

- Routing algorithms based non-Euclidean coordinate systems
  - VPCR (Newsome et al., 2003)
  - BVR(Fonseca et al., 2005)
  - S4(Mao et al., 2007)
- Do not scale as well as geographic routing for large (3,200 node) networks (SenSys 2010)

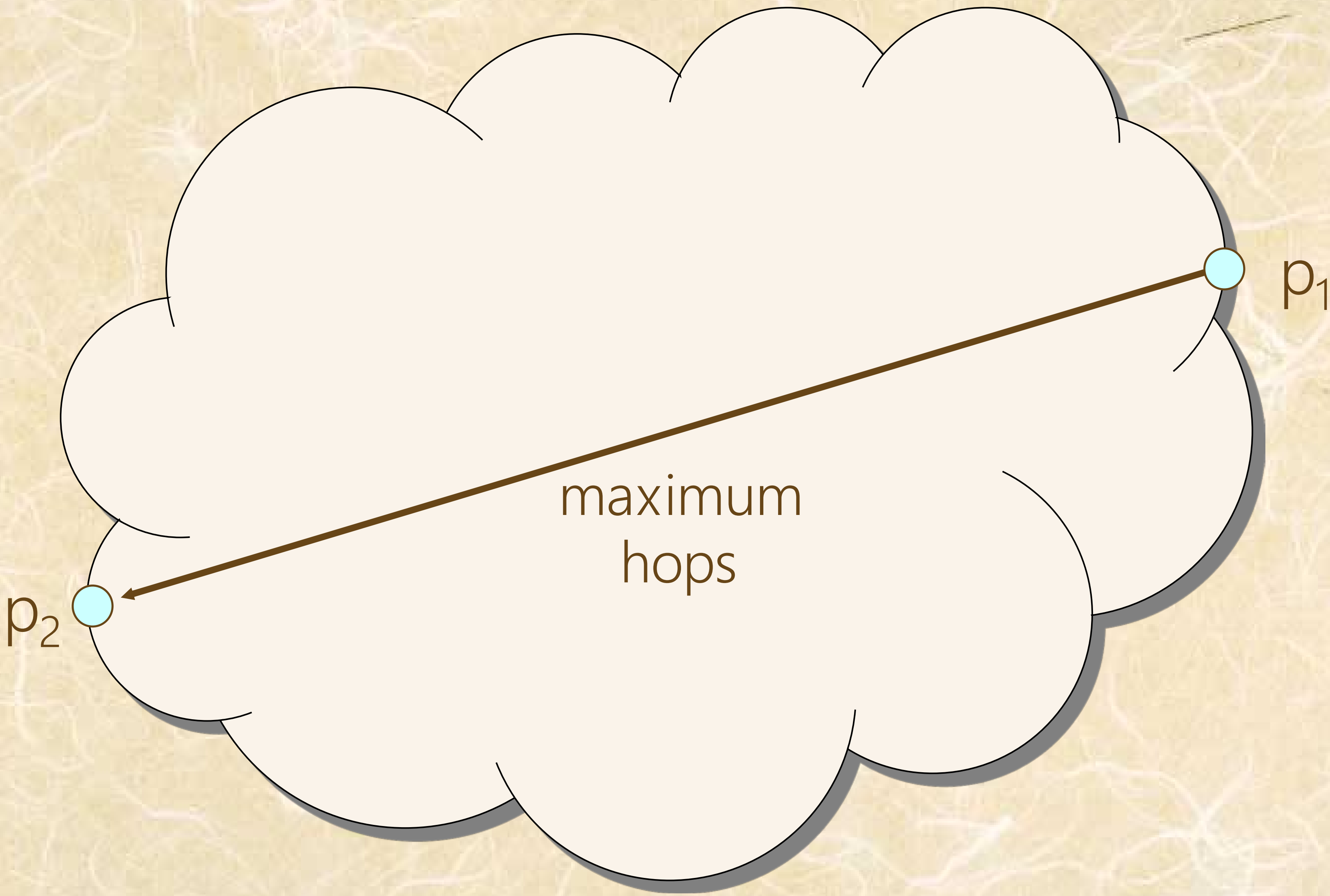
GSpring (Leong et al., 2007)



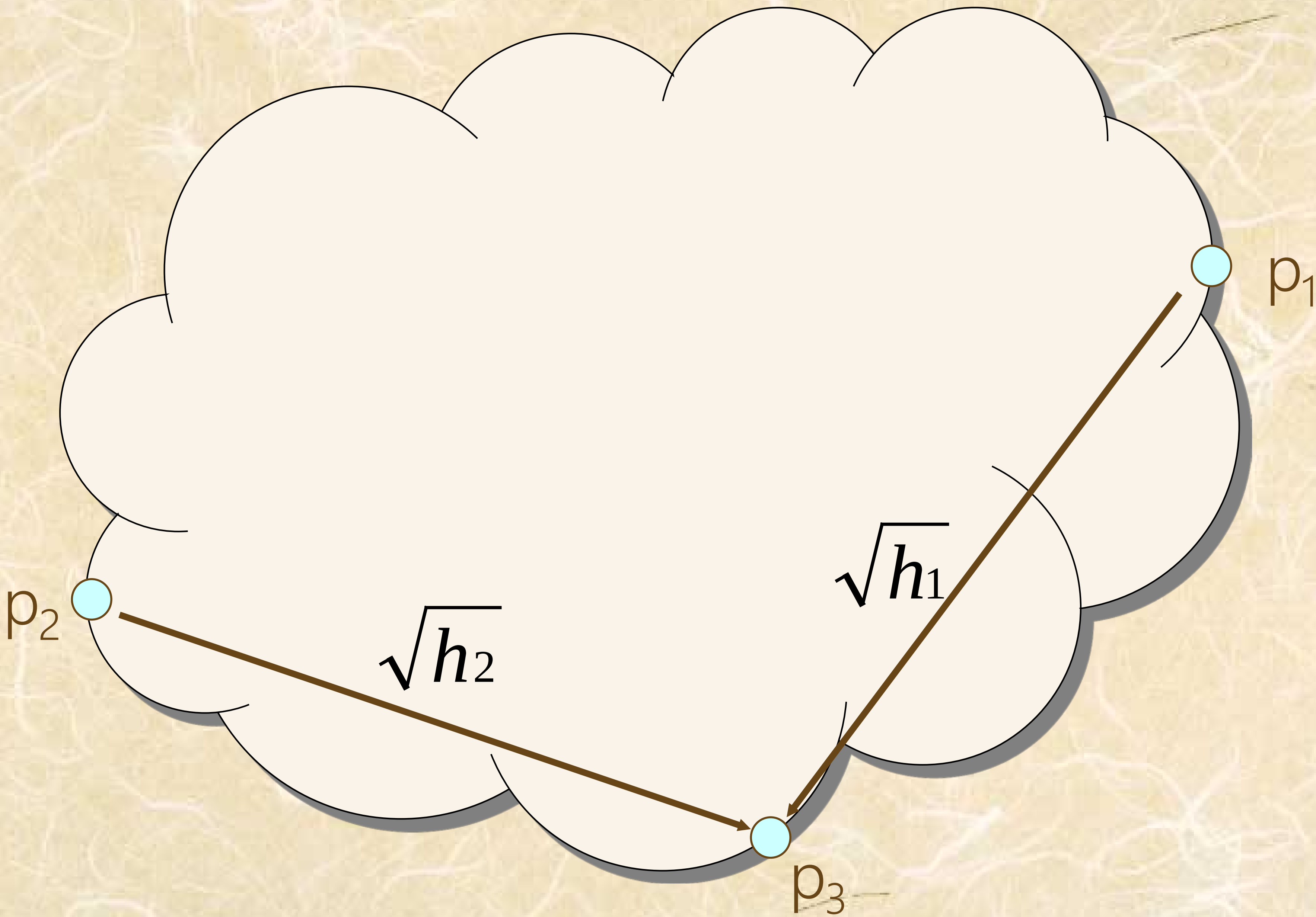
GSpring (Leong et al., 2007)



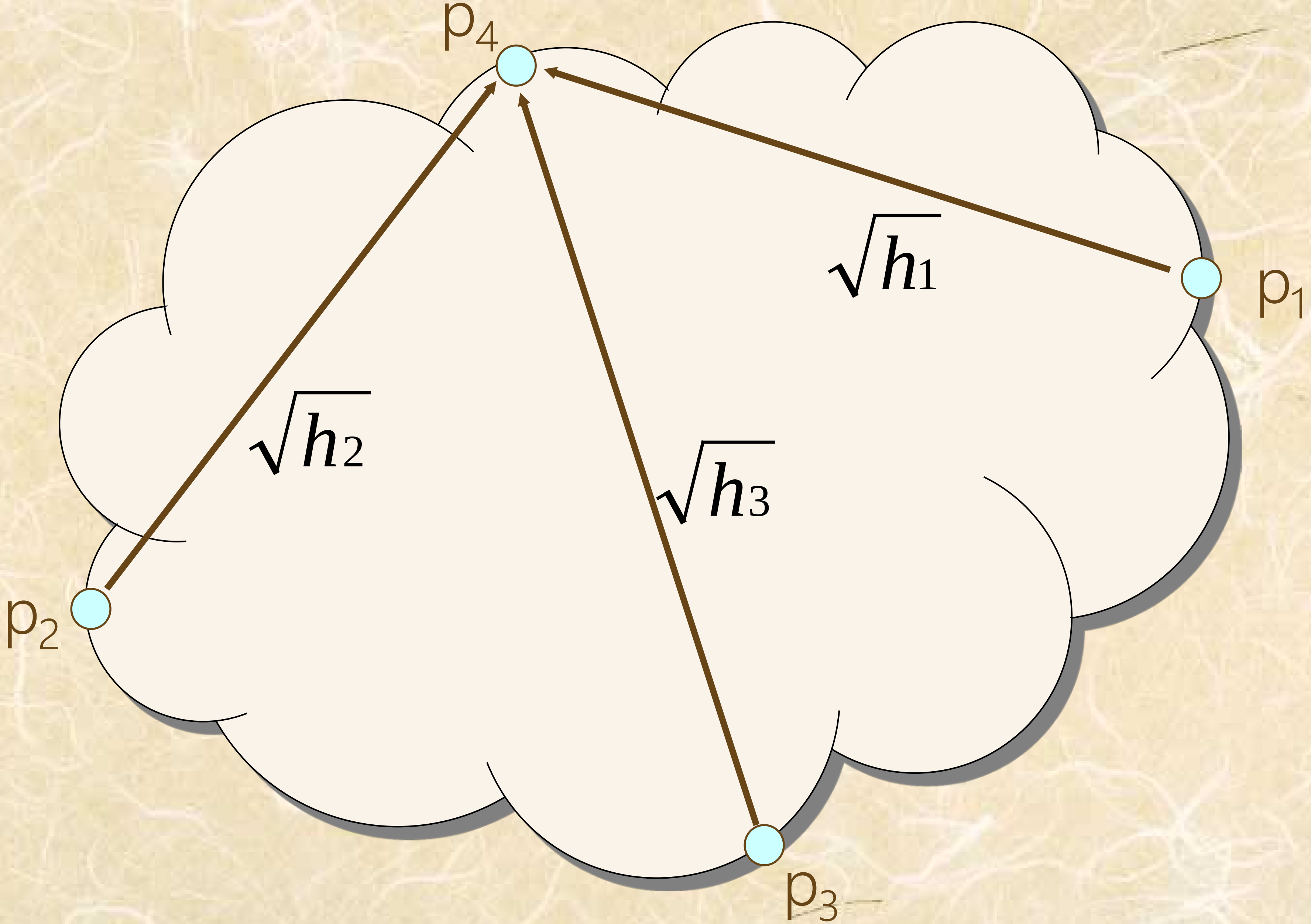
GSpring (Leong et al., 2007)



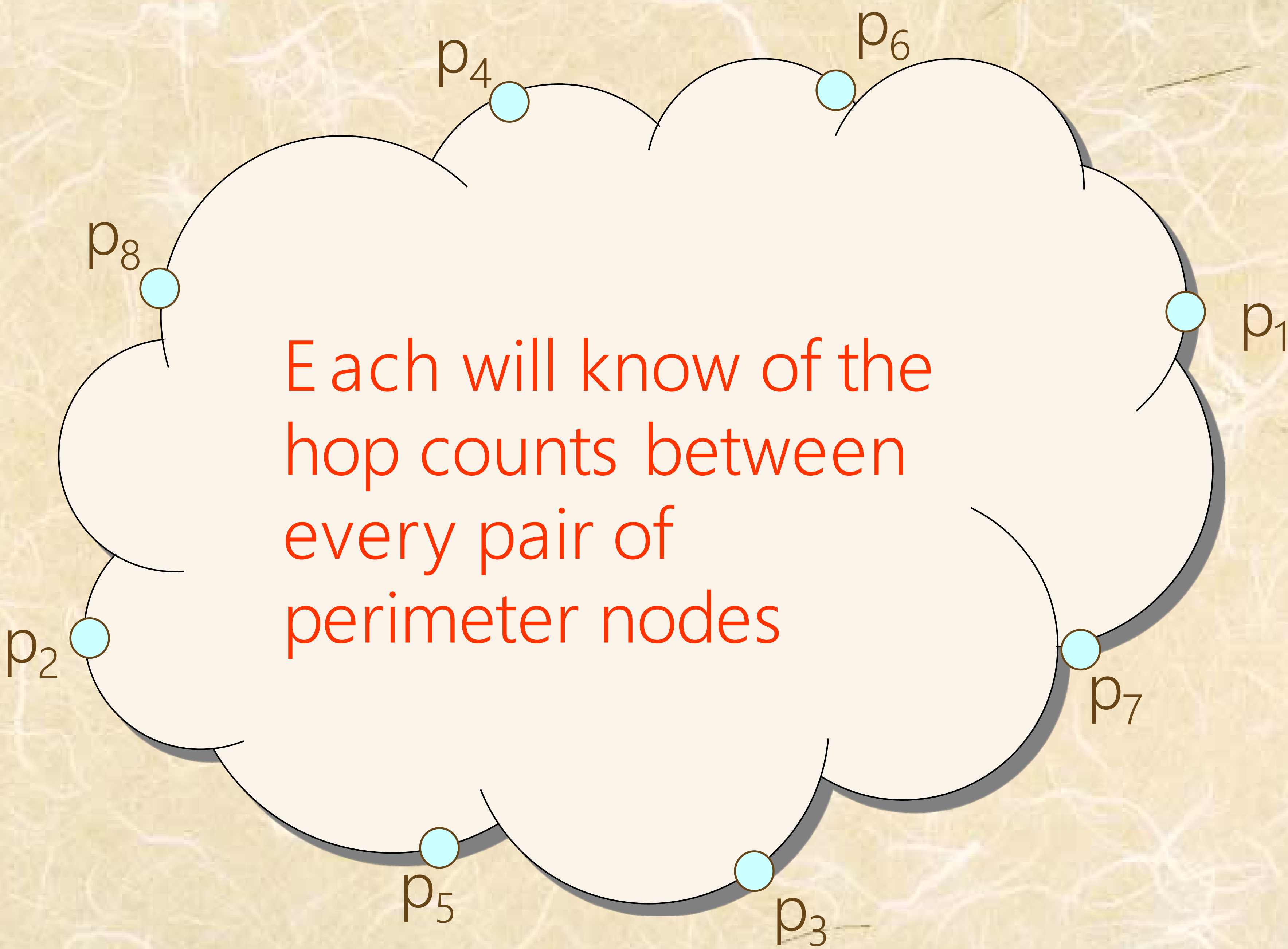
GSpring (Leong et al., 2007)



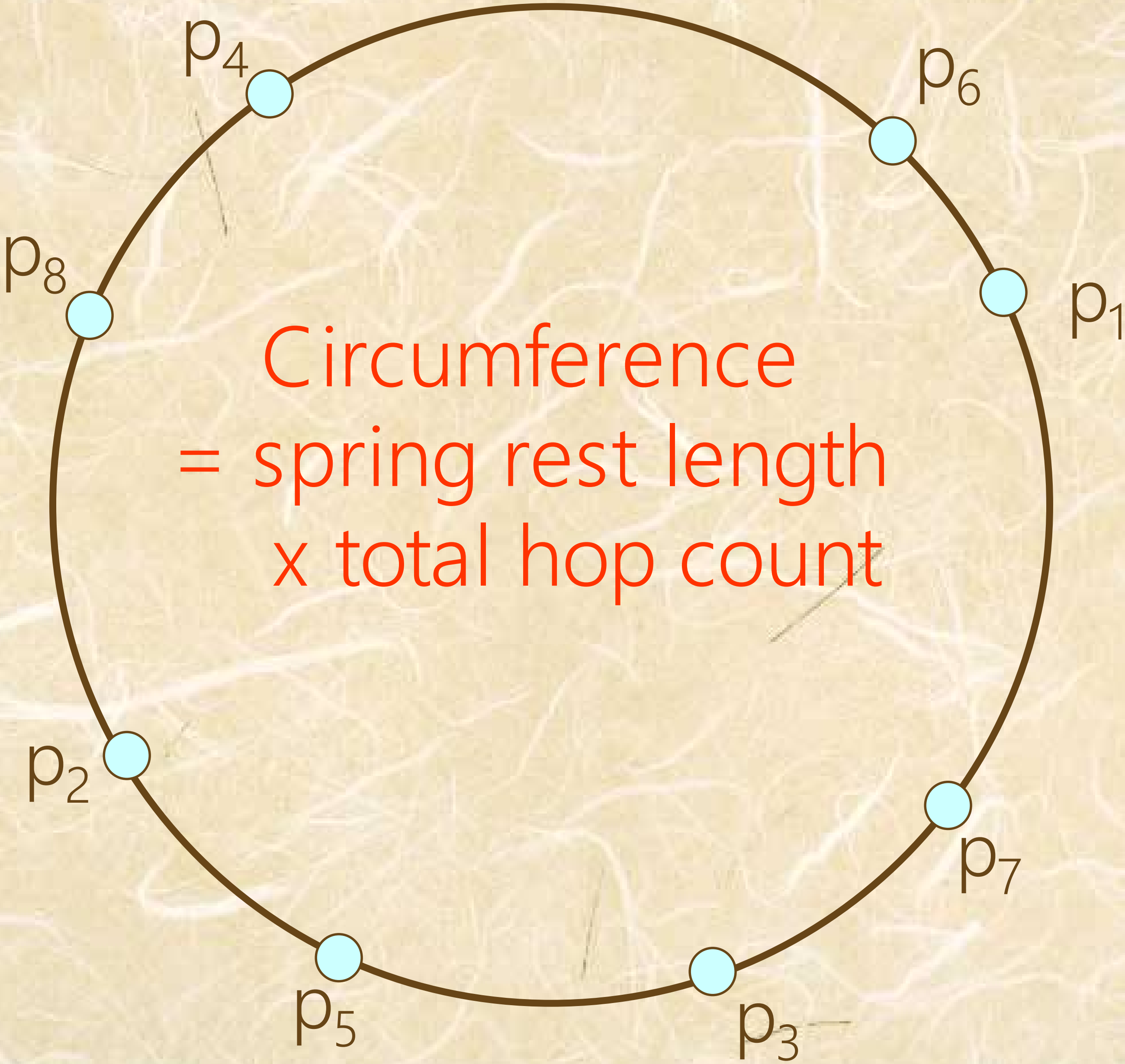
GSpring (Leong et al., 2007)



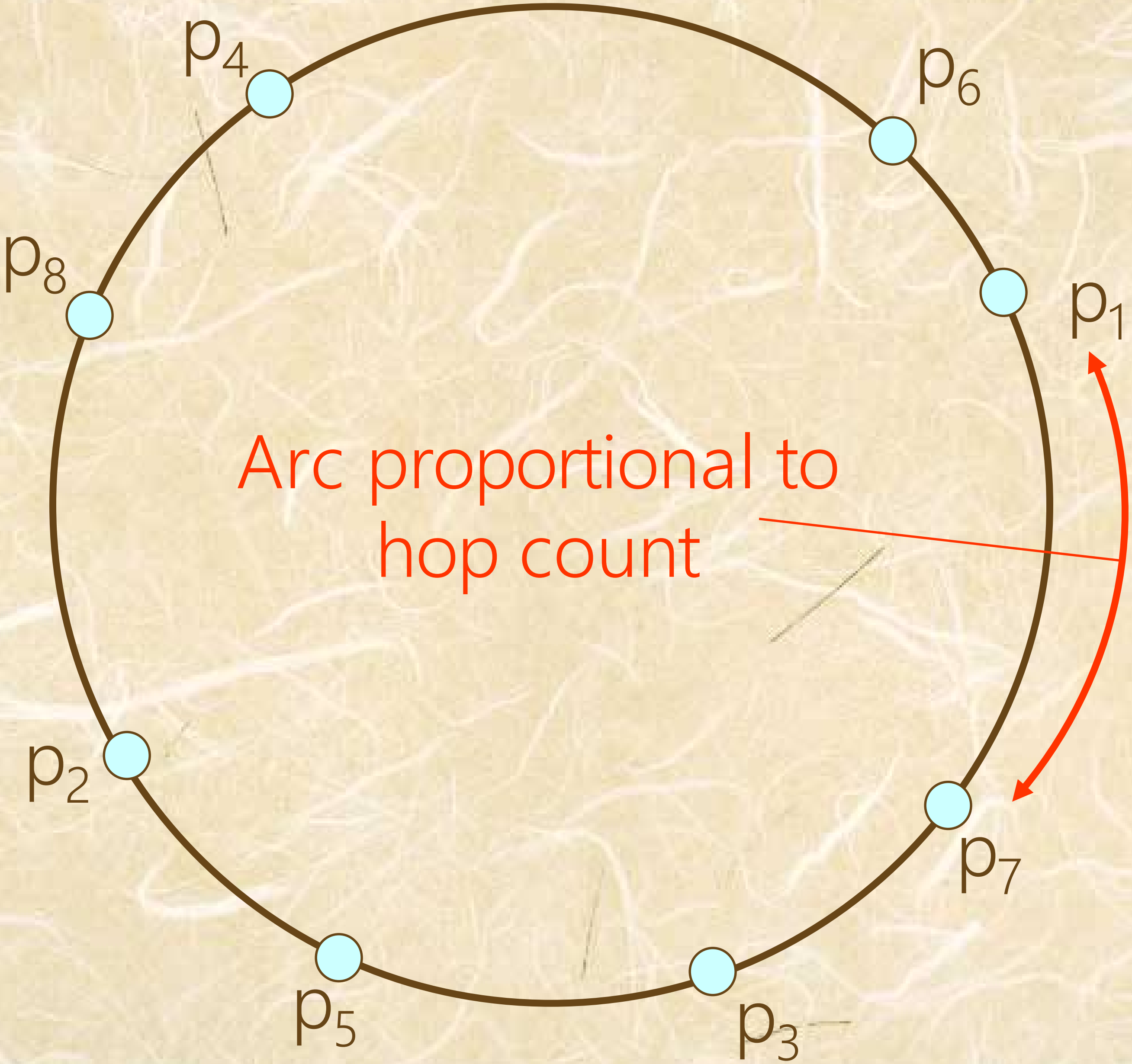
GSpring (Leong et al., 2007)



Projection onto Circle



Projection onto Circle



Particle Swarm Virtual Coordinates (PSVC)

- Based on hop count
- Reference nodes are elected to compute initial coordinates for all nodes
- Using PSO algorithm to minimize the error when computing initial coordinates
- Running a iterative relaxation procedure to make the virtual topology more convex
- PSVC can be trivially extended to 3D coordinates

Determining Initial Coordinates

- Select reference nodes
- Initialization of reference nodes
- Coordinates for Non-reference nodes

Determining Initial Coordinates

- Select reference nodes
- Initialization of reference nodes
- Coordinates for Non-reference nodes

Initialization of reference nodes

p_1 → (0,0)

p_2 → ($100h_{12}$,0) // h_{ij} is the hop count from p_i to p_j

p_3 → (x_3,y_3) // x_3 and y_3 are computed with triangle equalities using $100h_{31}$ and $100h_{32}$, while error function is as

$$E = \sum_{i=1}^{k-1} (|\overrightarrow{x_k} - \overrightarrow{x_i}| - 100h_{ik})^2$$

Determining Initial Coordinates

- Select reference nodes
- Initialization of reference nodes
- Coordinates for Non-reference nodes

Coordinates for Non-reference nodes

- Hop counts to all the reference nodes
- The coordinates of all the reference nodes
- Use PSO to minimize the error of objective function that maps the assigned virtual coordinates for each node to their hop counts to the reference nodes

PSO equation and parameters

$$\begin{cases} \overrightarrow{v_i} = w\overrightarrow{v_i} + c_1 r_1 (\overrightarrow{l_i} - \overrightarrow{x_i}) + c_2 r_2 (\overrightarrow{G_i} - \overrightarrow{x_i}) \\ \overrightarrow{x_i} = \overrightarrow{x_i} + \overrightarrow{v_i} \\ w = w_{\max} - \dfrac{k}{k_{\max}} (w_{\max} - w_{\min}) \end{cases}$$

Parameters:

$POPSIZE = 10, w_{\max} = 1.2, w_{\min} = 0.1, k_{\max} = 100, c_1 = c_2 = 1.8,$

$r_1, r_2 \in [0,1]$

Notes: To prevent floating point overflow, all inputs are normalized by dividing them by $100h_{12}$

Algorithm1: Compute initial coordinates for non-reference nods with PSO

Given: $\overrightarrow{p_i}, i = 1, \cdots, p$

Initialize $\overrightarrow{x_i} \in [-1,1], \overrightarrow{v_i} \in [-1,1], \overrightarrow{l_i} = \vec{0}, i = 1, \cdots, POPSIZE$

$Lerror_i = \infty, i = 1, \cdots, POPSIZE, \overrightarrow{G_i} = \vec{0}, Gerror = \infty$

for k=0 to $k_{\max}$ do

$w = w_{\max} - \frac{k}{k_{\max}}(w_{\max} - w_{\min})$

for i=0 to *POPSIZE* do

$\overrightarrow{v_i} \leftarrow w\overrightarrow{v_i} + c_1 r_1(\overrightarrow{l_i} - \overrightarrow{x_i}) + c_2 r_2(\overrightarrow{G_i} - \overrightarrow{x_i})$

$\overrightarrow{x_i} \leftarrow \overrightarrow{x_i} + \overrightarrow{v_i}$

$error = \sum_{j=1}^{p}(|\overrightarrow{x_i} - \overrightarrow{p_j}| - h_j / h_{ij})^2$

if $error < Lerror_i$ then

$\overrightarrow{l_i} \leftarrow \overrightarrow{x_i}$

$Lerror_i = error$

end if

if $error < Gerror$ then

$\vec{G} \leftarrow \overrightarrow{x_i}$

$Gerror = error$

end if

end for

end for

// where p is the number of the current node to the reference node j, $\overrightarrow{p_j}$ is the position for reference node j.

Relaxation after initialization

- Spring force:

$$\overrightarrow{F_{ij}} = \kappa \times (l_{ij} - |\overrightarrow{x_i} - \overrightarrow{x_j}|) \times u(\overrightarrow{x_i} - \overrightarrow{x_j})$$

(Hooke's Law)

- Net force:

$$\overrightarrow{F_i} = \sum_{j \neq i} \overrightarrow{F_{ij}}$$

- Update rule:

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \frac{\min(|\overrightarrow{F_i}|, \alpha_t)}{|\overrightarrow{F_i}|} \overrightarrow{F_i}$$

Node joins after Convergence

- Listens to the stayalive beacons of its neighbors to obtain their coordinates
- If all its neighbors have stabilized, it computes its coordinates as a weighted sum of the coordinates of its neighbors' coordinates

$$x_i = \frac{1}{\sum_j r_{ij}} \sum_j r_{ij} x_j$$

$$y_i = \frac{1}{\sum_j r_{ij}} \sum_j r_{ij} y_j$$

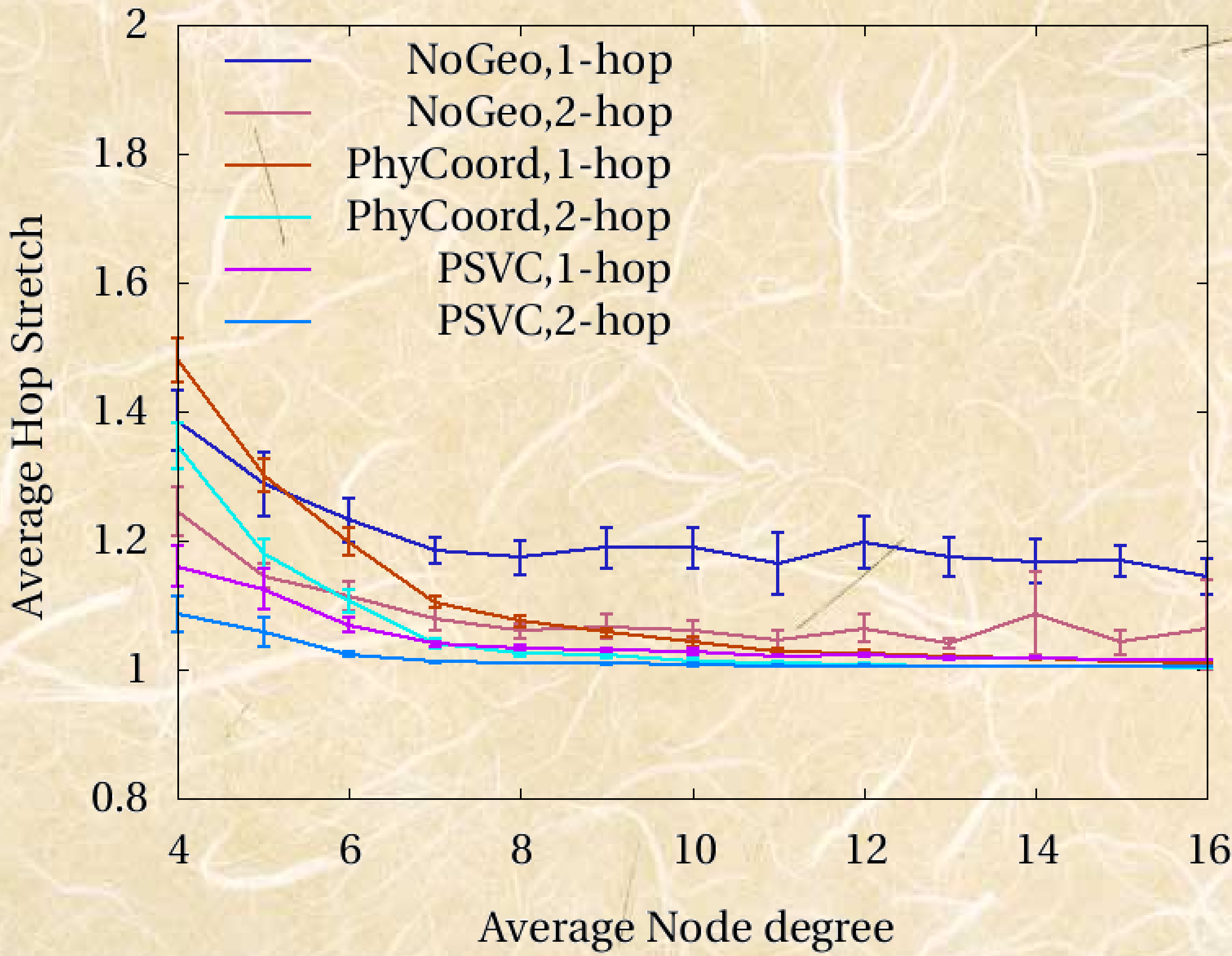
Performance

- Evaluation
  - Testbed
  - TinyOS Simulator
- Measured metrics:
  - Greedy forwarding success rate
  - Hop Stretch
  - Storage cost
  - Overhead
- Simulation topologies
  - range of network densities (average node degree)
  - larger networks up to 3,200 nodes
    - low/high density
    - obstacles

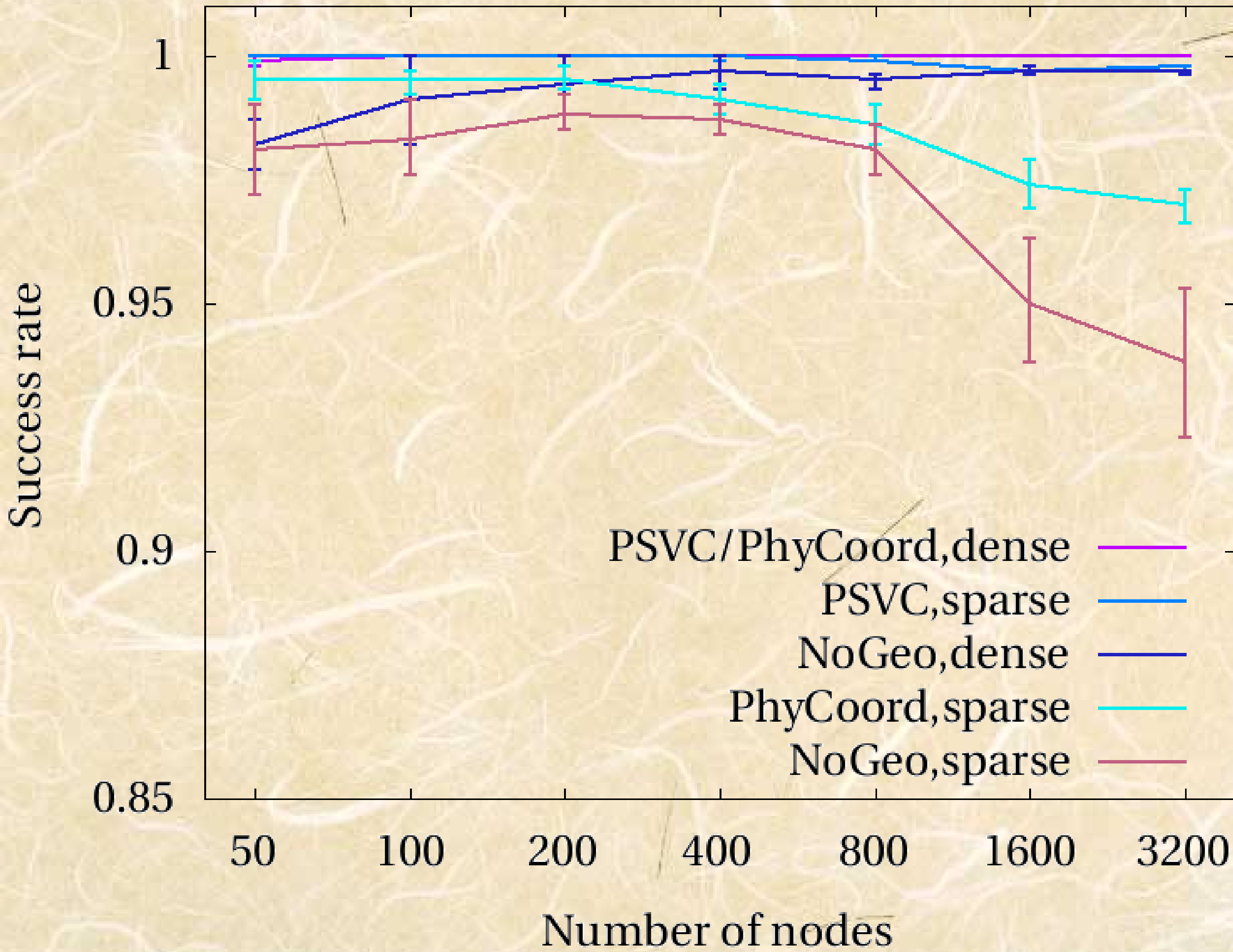
Performance

- Routing algorithm: GDSTR
- Compare with
  - actual coordinates
  - NoGeo (Rao et al., 2003)

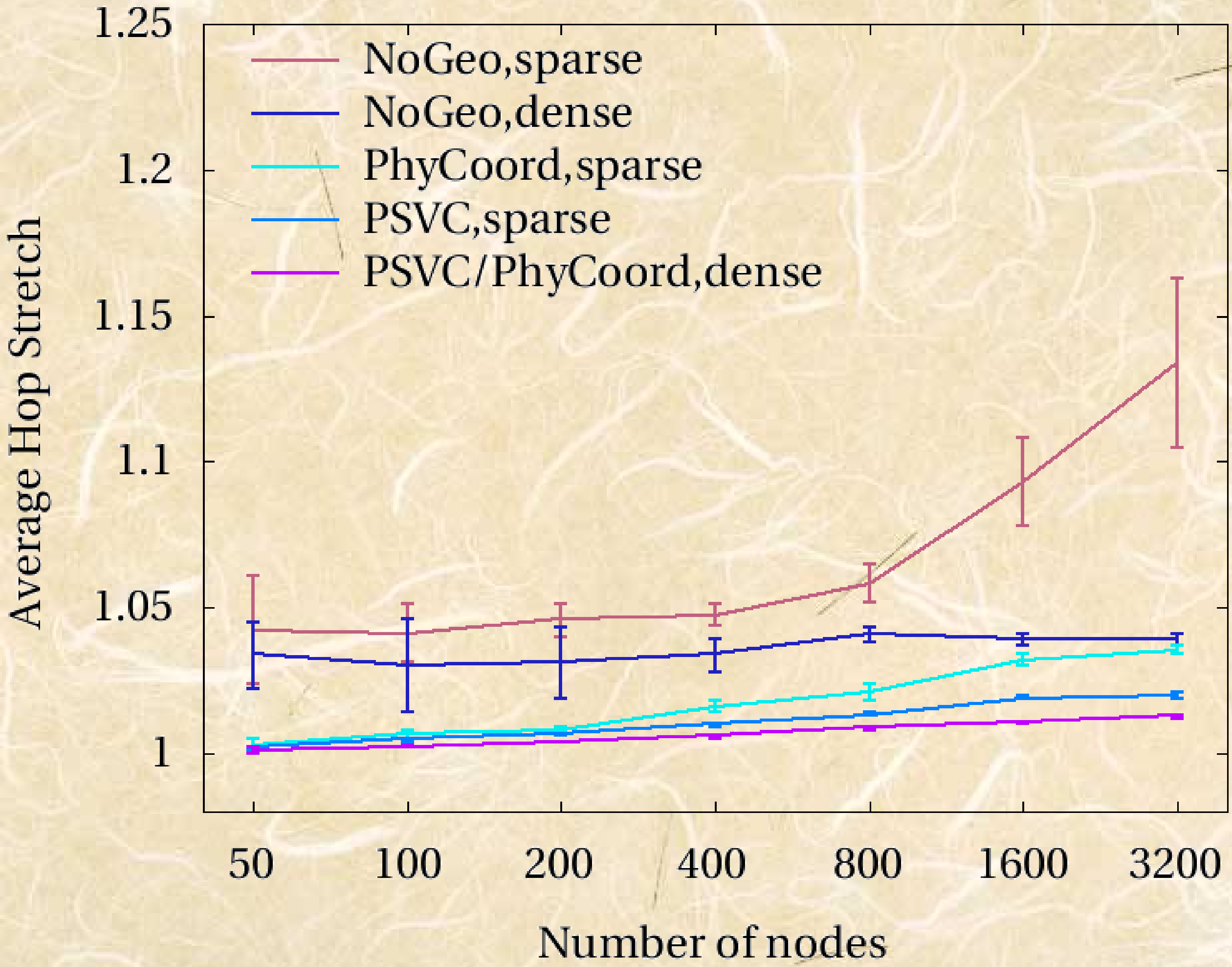
Average hop stretch for GDSTR against network density



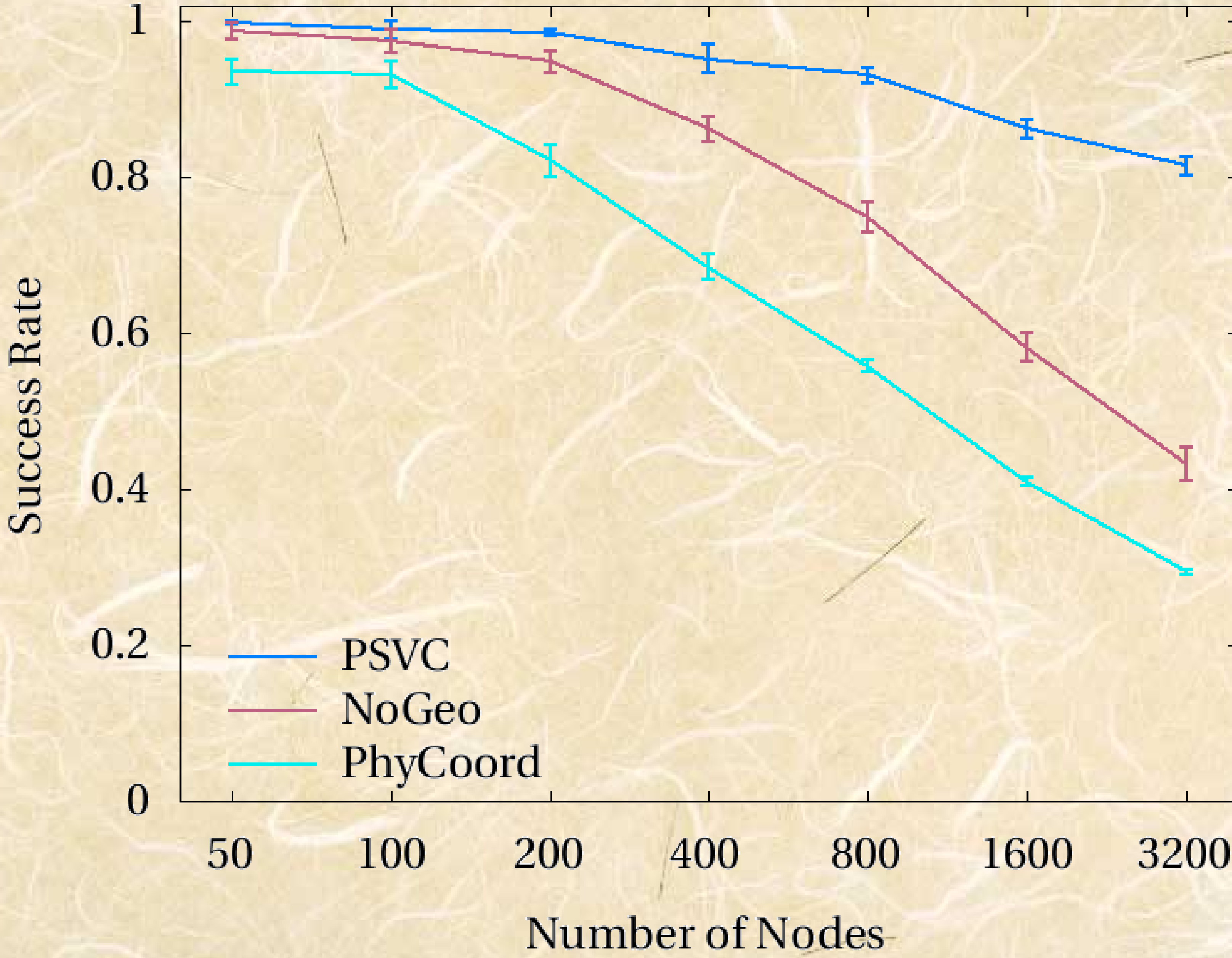
Two-hop greedy forwarding success rate against network size in 2D networks



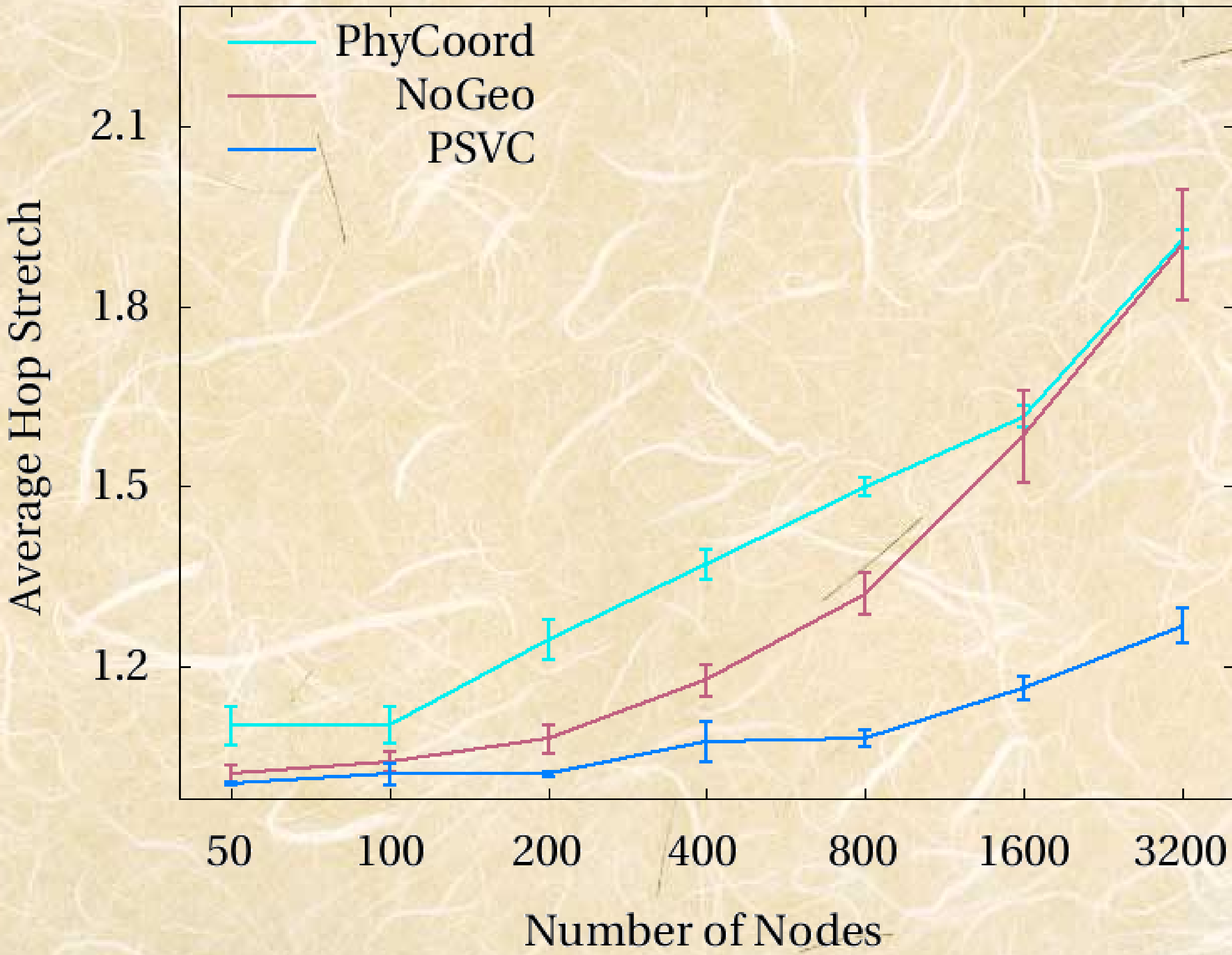
Average hop stretch against network size in 2D networks



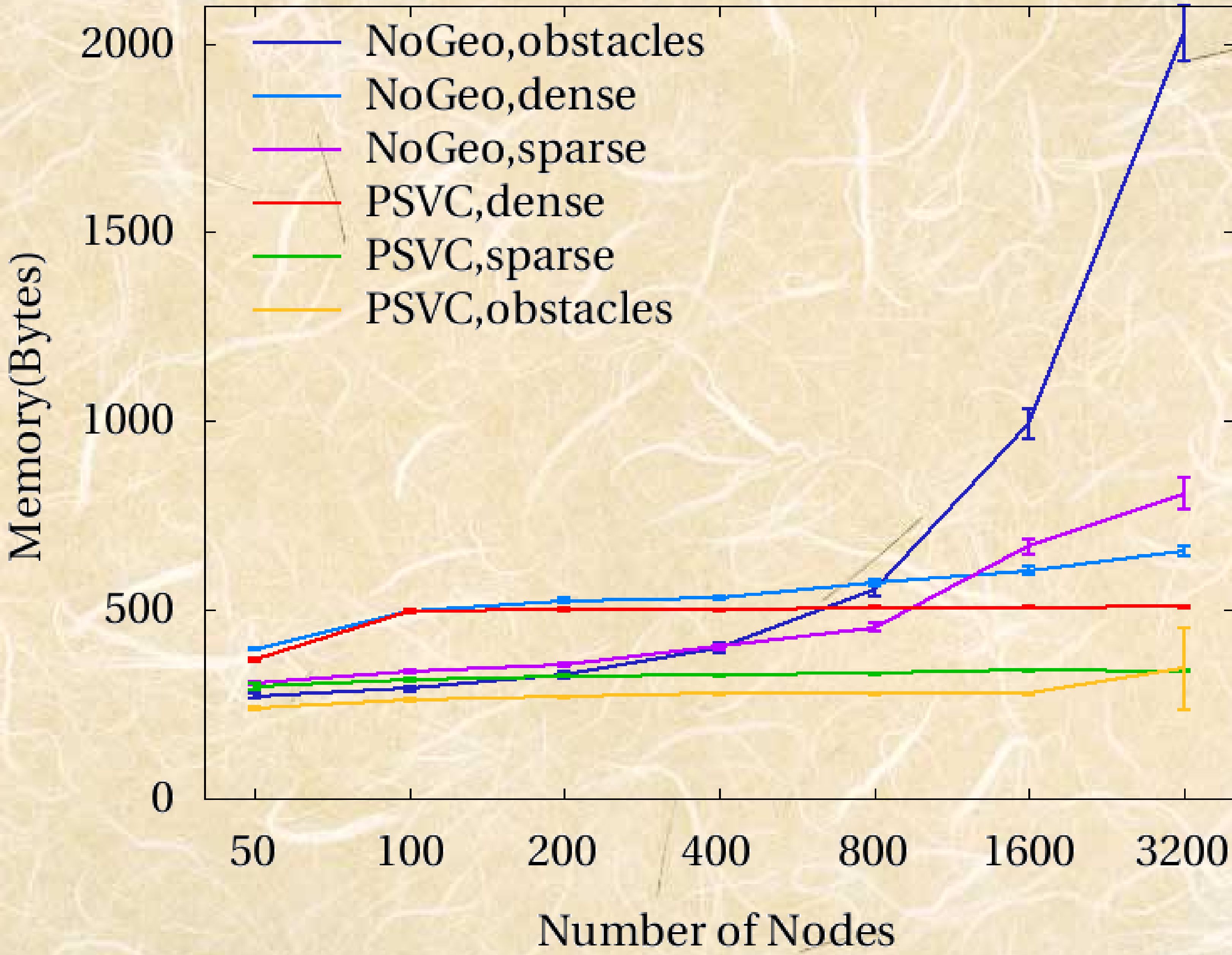
Two-hop greedy forwarding success rate against network size for 2D networks with obstacles



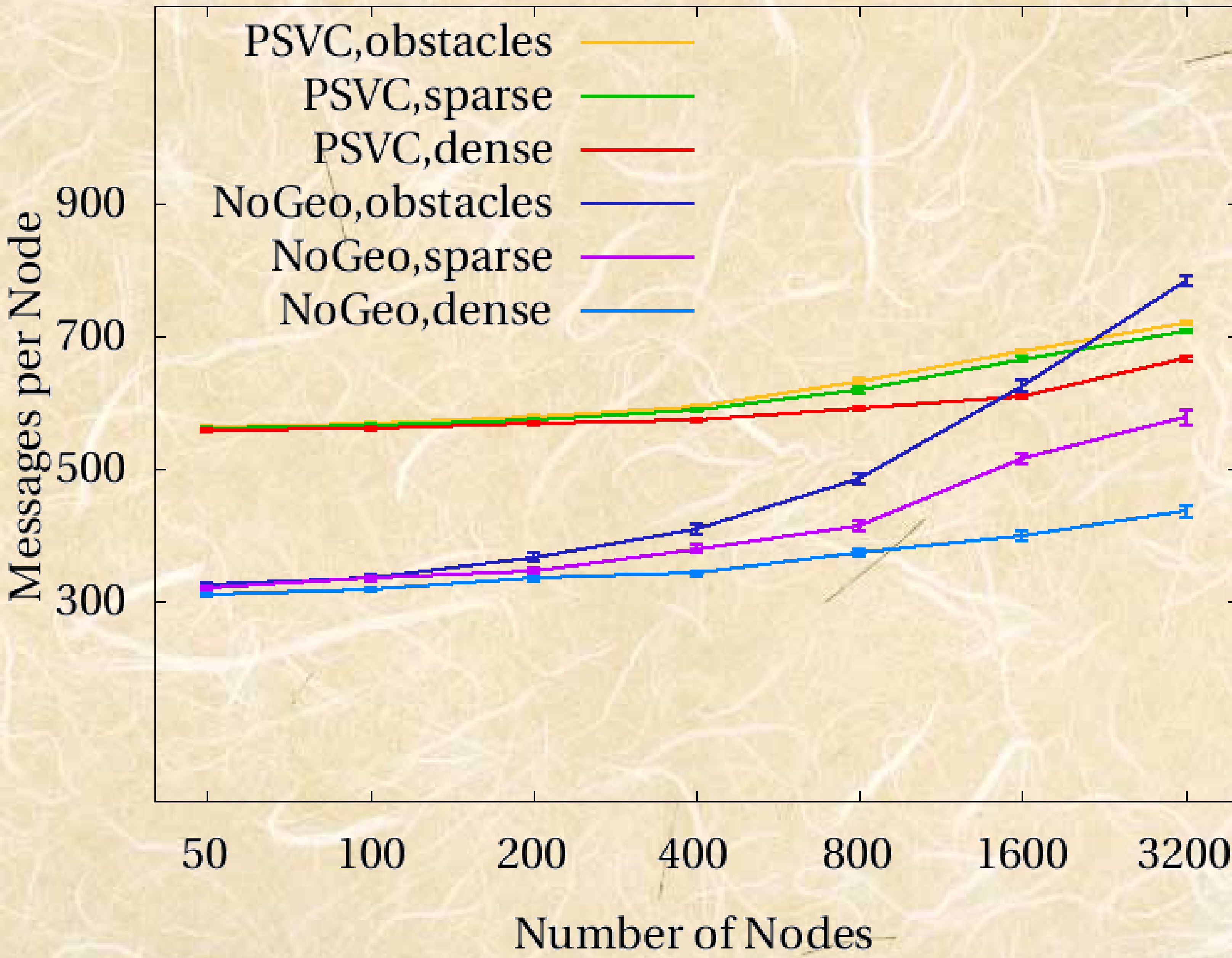
Average hop stretch for GDSTR against network size for 2D with obstacles



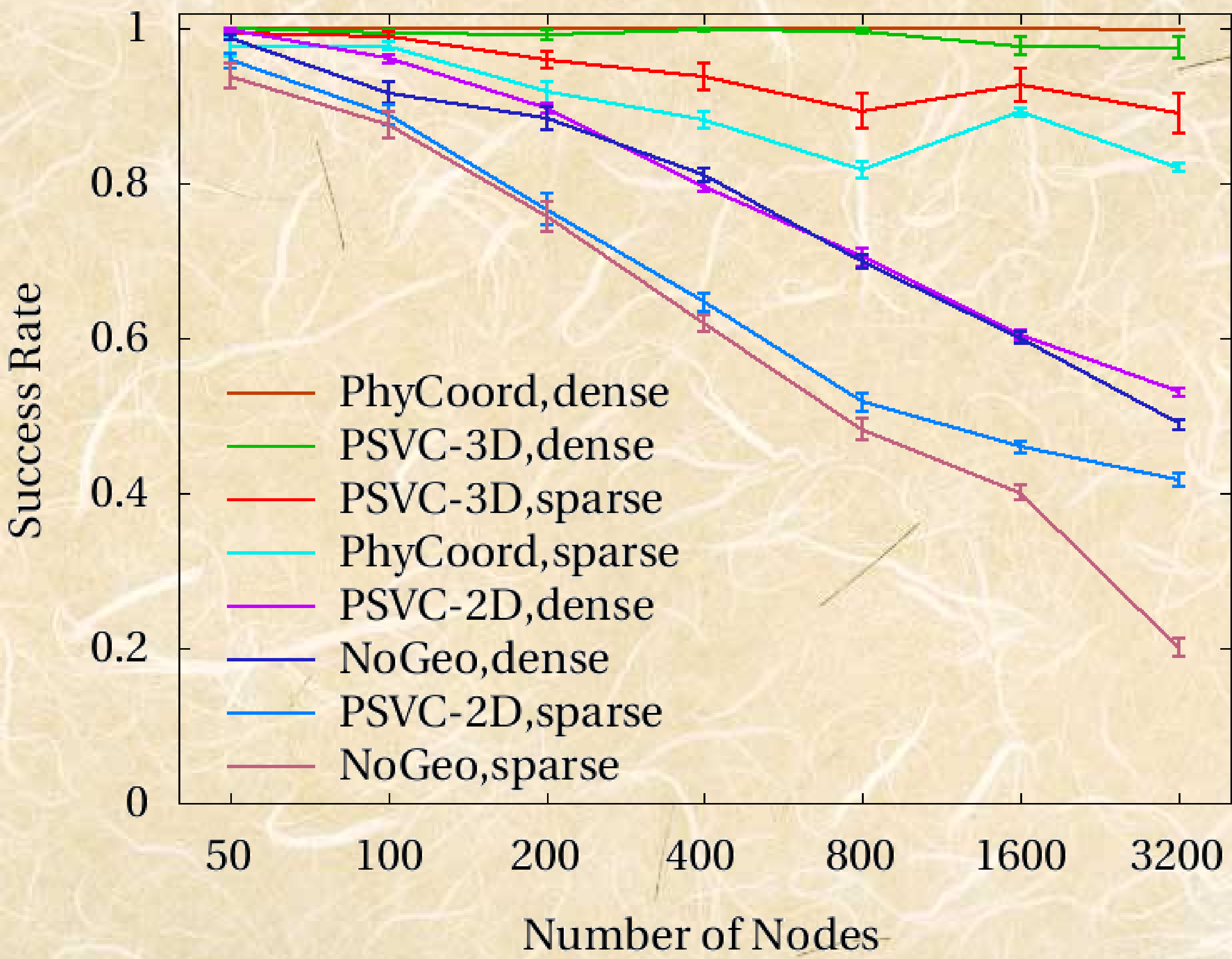
Maximum storage cost versus network size for 2d networks



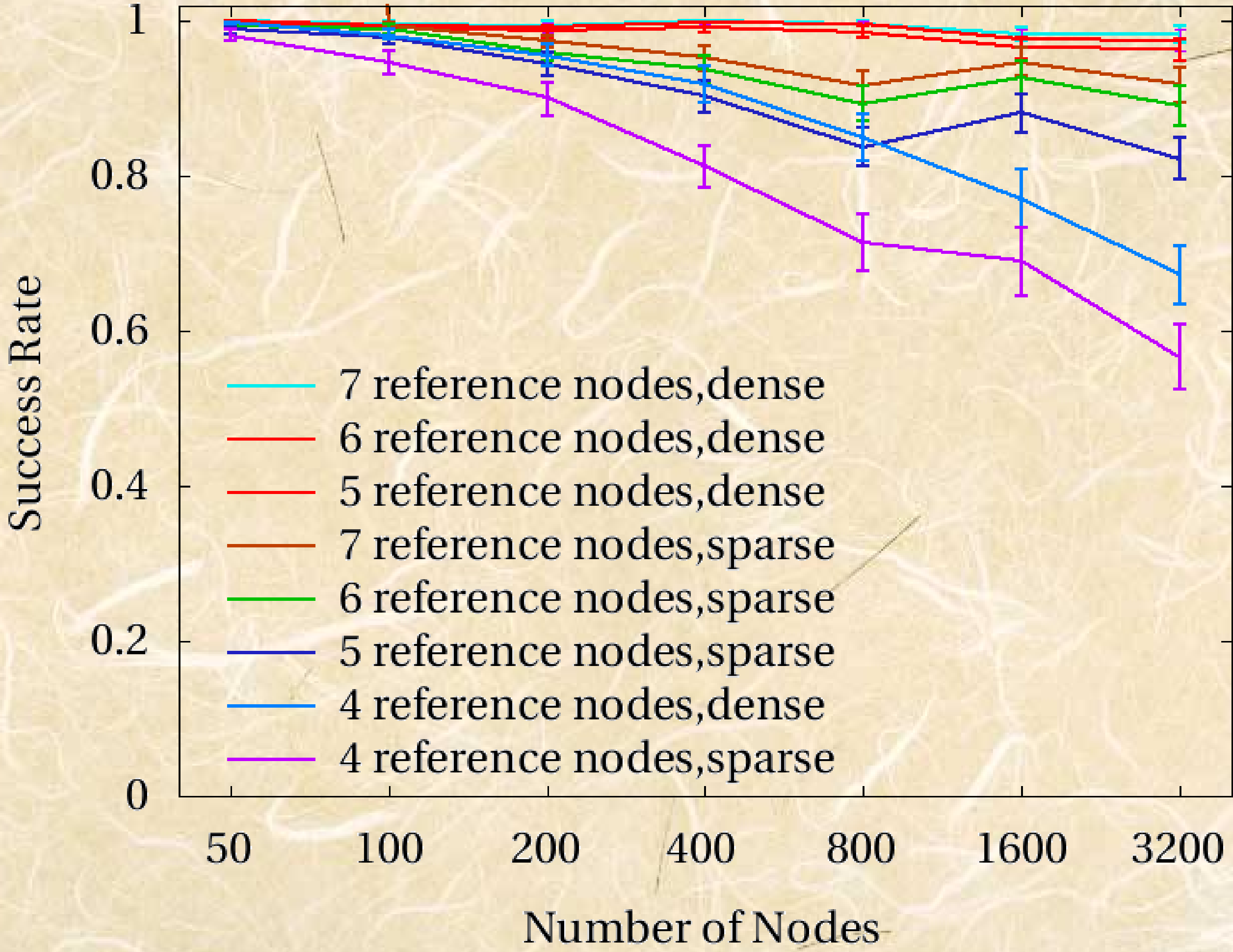
Messages sent per node against network size for 2D networks



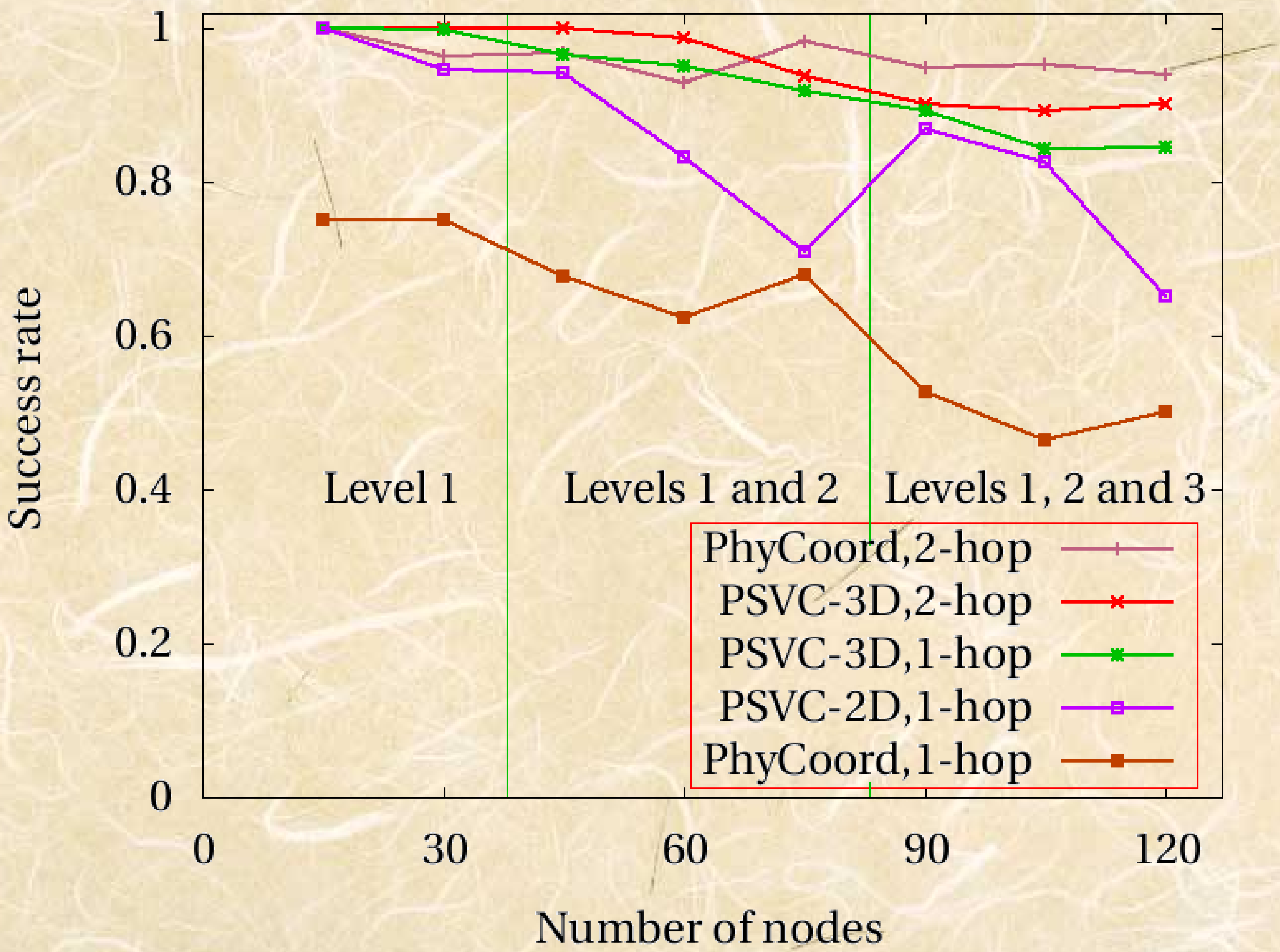
Two-hop greedy success rate against network size for 3D networks



Greedy forwarding success rate for PSVC with 4, 5, 6 and 7 reference nodes for 3D networks



Two-hop greedy forwarding success rate for various algorithms on Indriya



Conclusion

- Converges faster and a**chieves a lower hop stretch compared to NoGeo**
- Scales well up to 3,200 nodes
- Makes no assumptions on the network topology and can naturally be extended to three-dimensional(3D) wireless sensor networks