Iso-contour Queries and Gradient Descent With Guaranteed Delivery in Sensor Networks

Rik Sarkar **Xianjin Zhu** **Jie Gao**

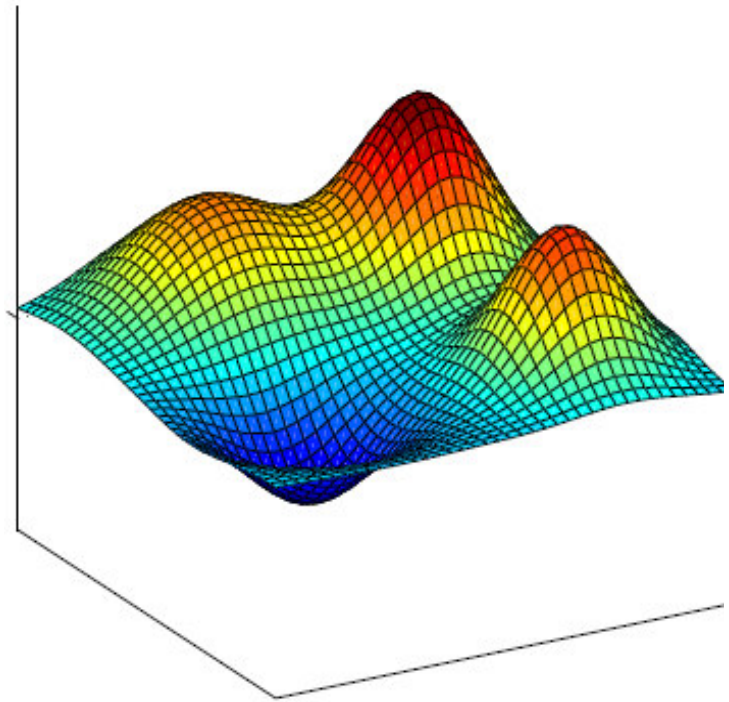
Dept. of Computer Science, Stony Brook University

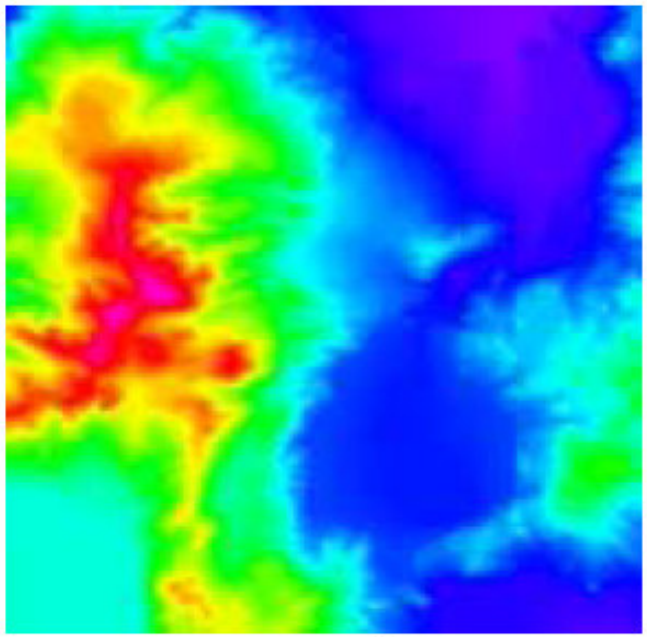
Leonidas Guibas
Dept. of Computer Science
Stanford University

Joseph Mitchell
Dept. of Applied Maths & Stats
Stony Brook University

Sensing and reacting to a spatial signal

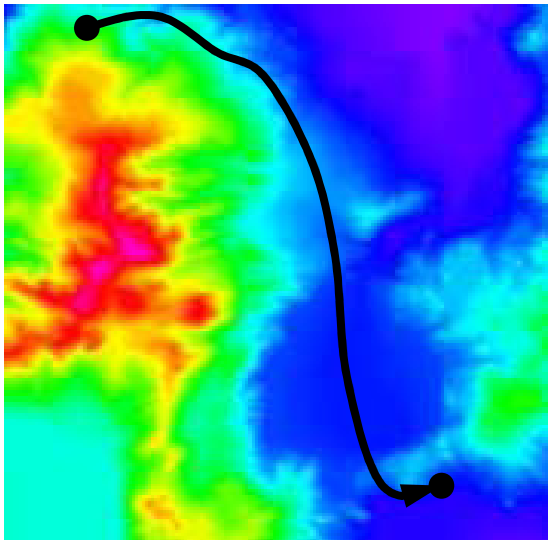
- Dense sensors distributed to monitor a spatial signal (temperature, traffic density, pollution level, etc).



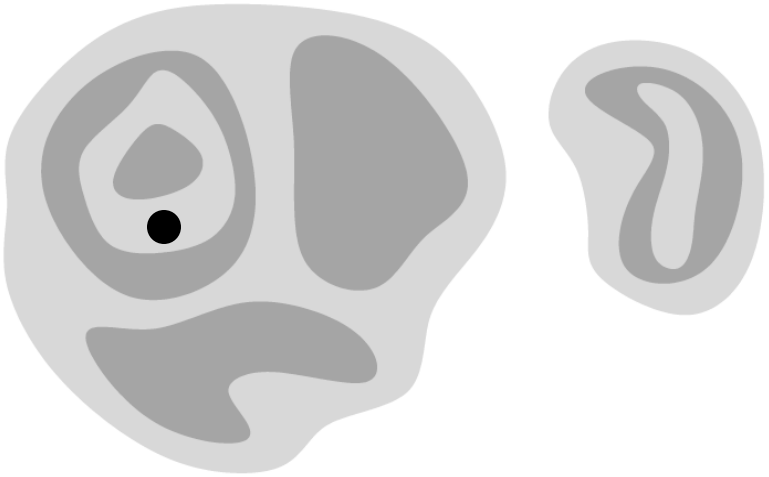


“A smart environment”

- Sensors monitor a spatial signal (temperature, traffic density, pollution level, etc).
- Distributed situation understanding
- Information-guided navigation
  - Real-time response and emergency rescue
  - Avoid traffic congestions



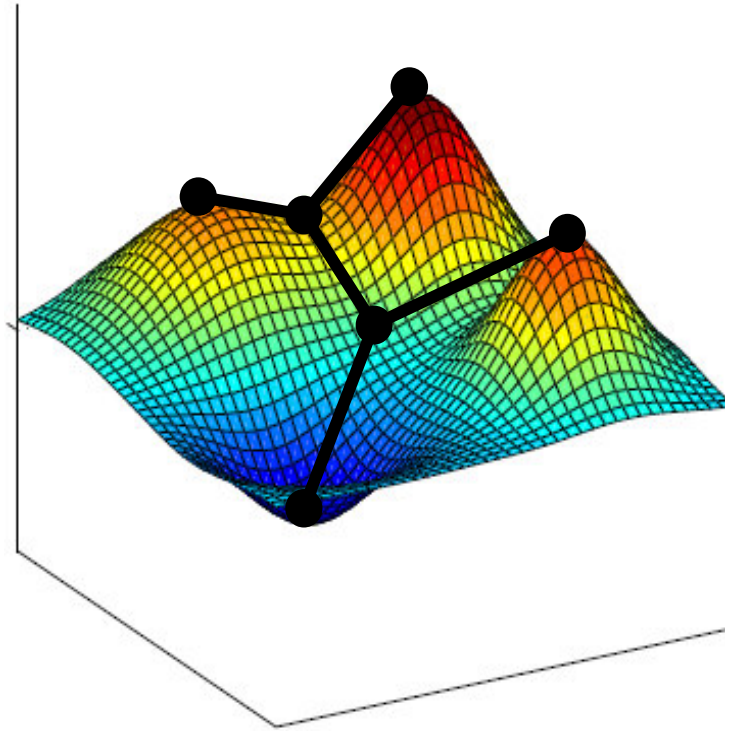
Find a low-value path?

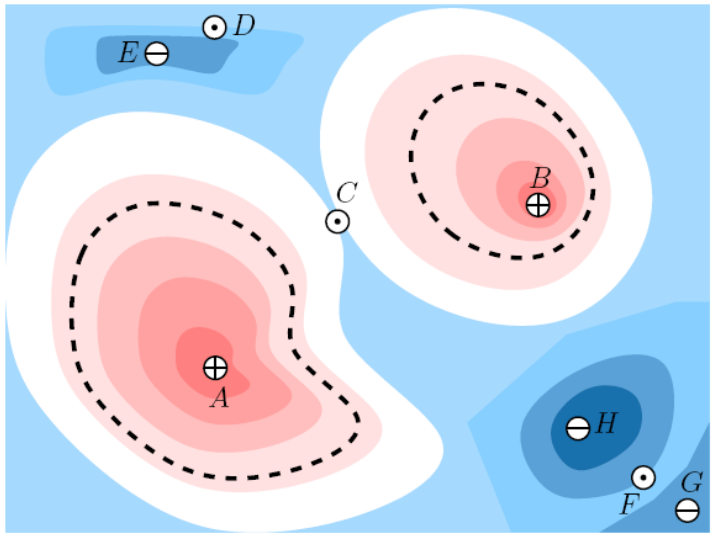


Am I surrounded?

Goal: find the spatial structure

- Find the contours
- Find local max/min, saddles, and relationship between them.



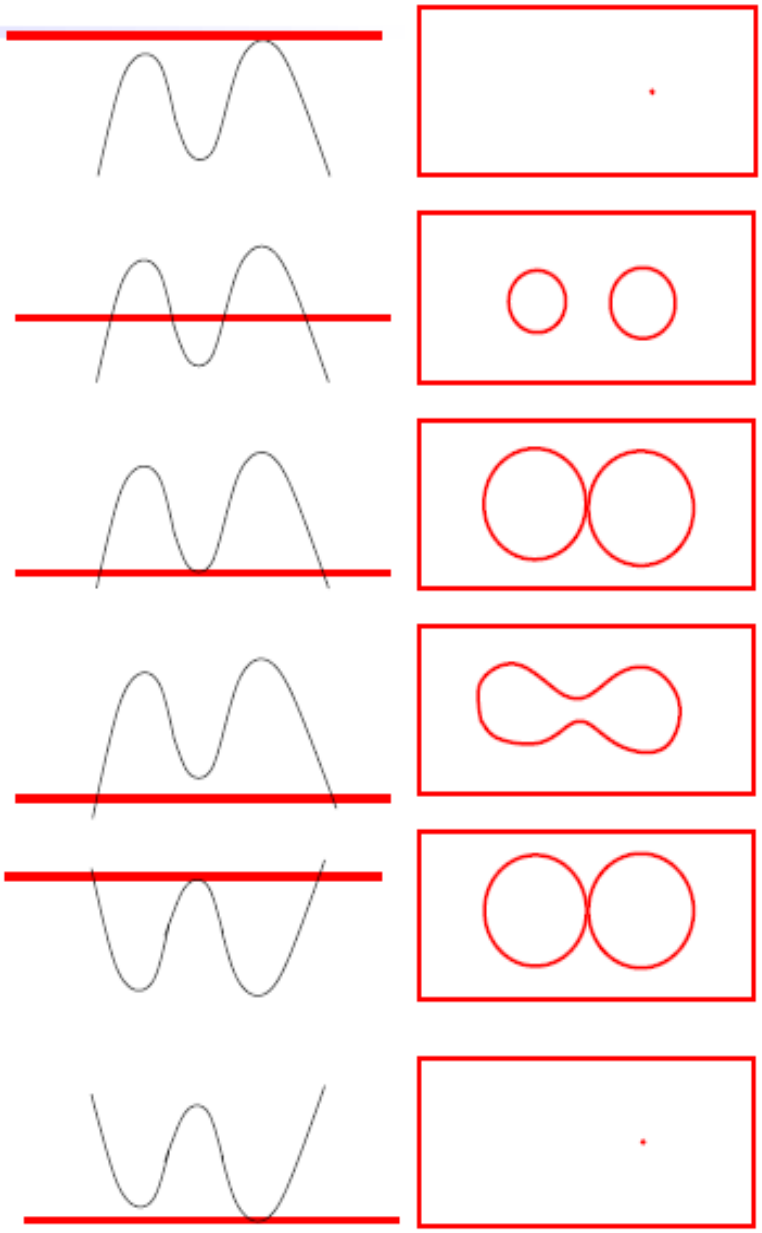


⊕ local maximum ⊖ local minimum

⊙ saddle point

Contour tree

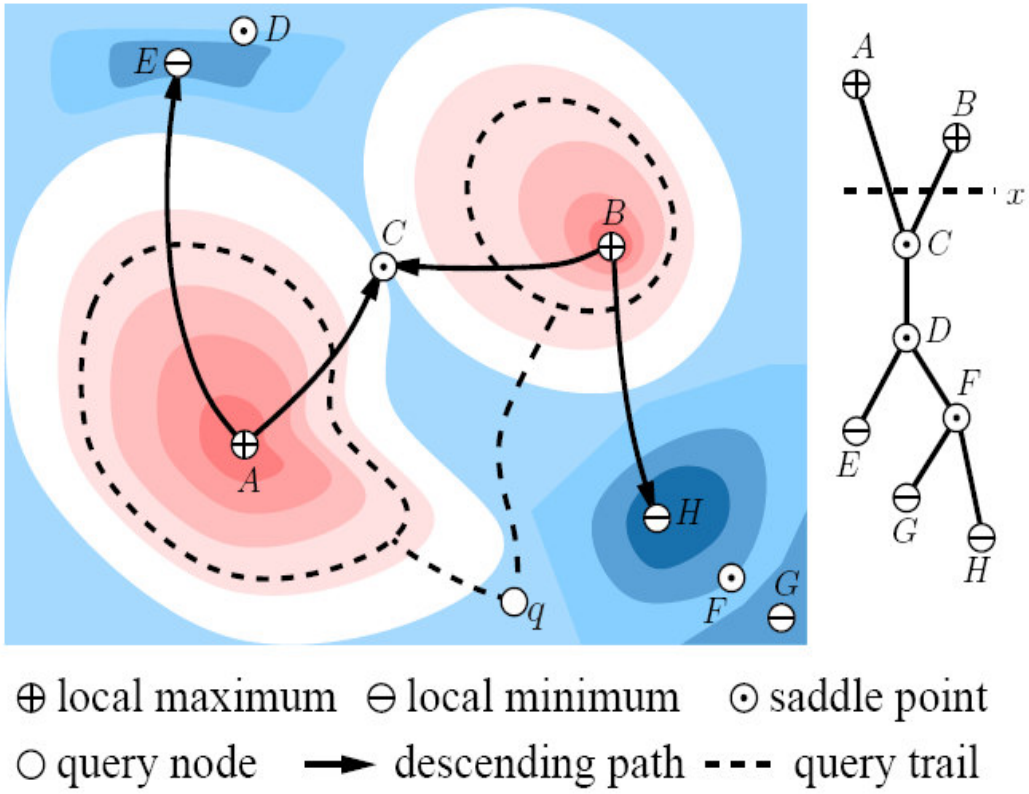
- A tree on critical points (local max/min, saddles).
- Capture the contour merging/splitting, emergence, disappearance.



Information guided navigation

- **Iso-contour queries**: find the contours at level x.
- **Low-value routing**: find a path from s to t with value below x.

Gradient descent methods + contour tree computation.

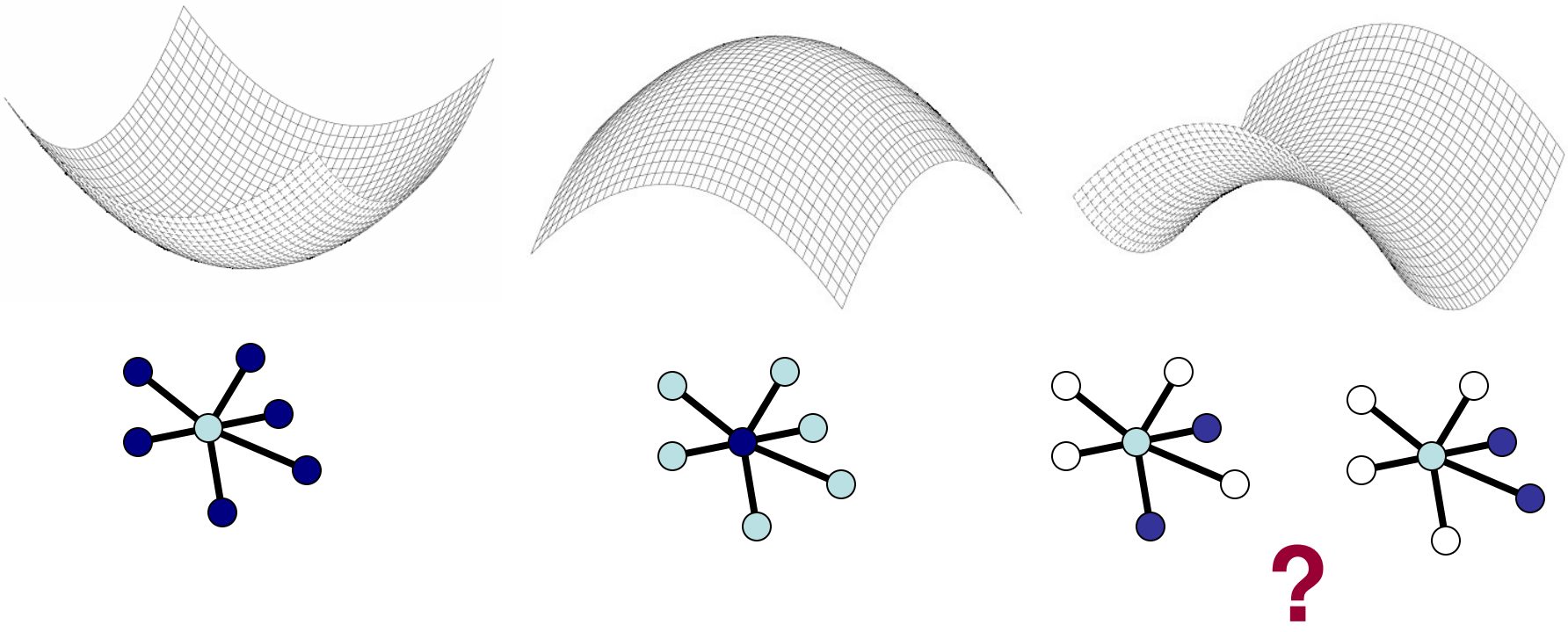


Centralized contour tree

- Efficient algorithms are known
- Uses global sorted order to sweep
- Uses locations, simplicial meshes, Union Find data structures
- References :
  - Kreveld, Oostrum, Bajaj, Pascucci, Schikore : SOCG ’97
  - Tarasov, Vyalyi : SOCG ’98
  - Carr, Snoeyink, Axen : SODA 2000
  - J. Milnor : Morse Theory

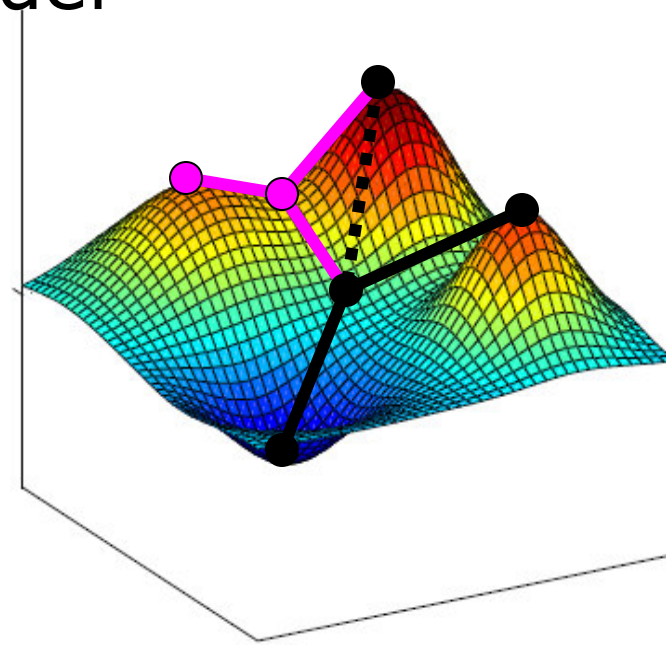
Challenges of computing a contour tree in a Sensor Net

1. No location or with location errors.
   - Can we locally identify max/min and saddles?



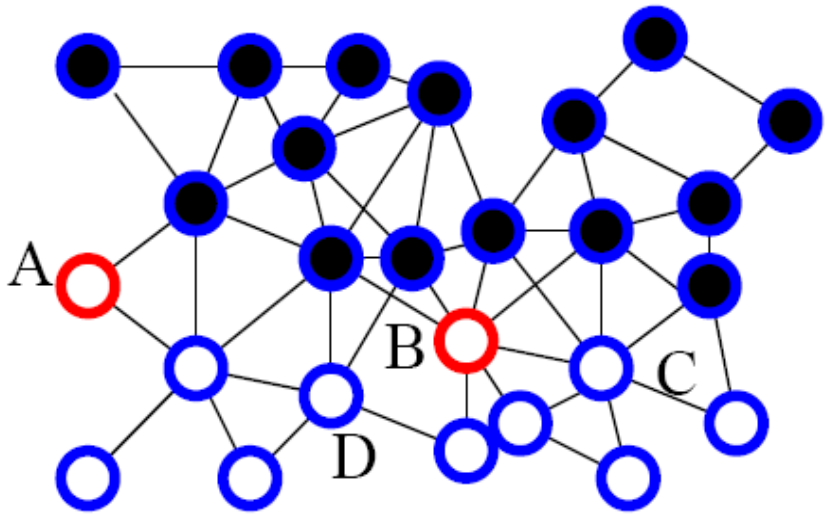
Challenges of computing a contour tree in a Sensor Net

1. No location or inaccurate locations.
2. Hard to detect *Saddles*
3. Distributed Environment
4. No globally sorted order
   - Hard to *sweep*



Use distributed sweeps

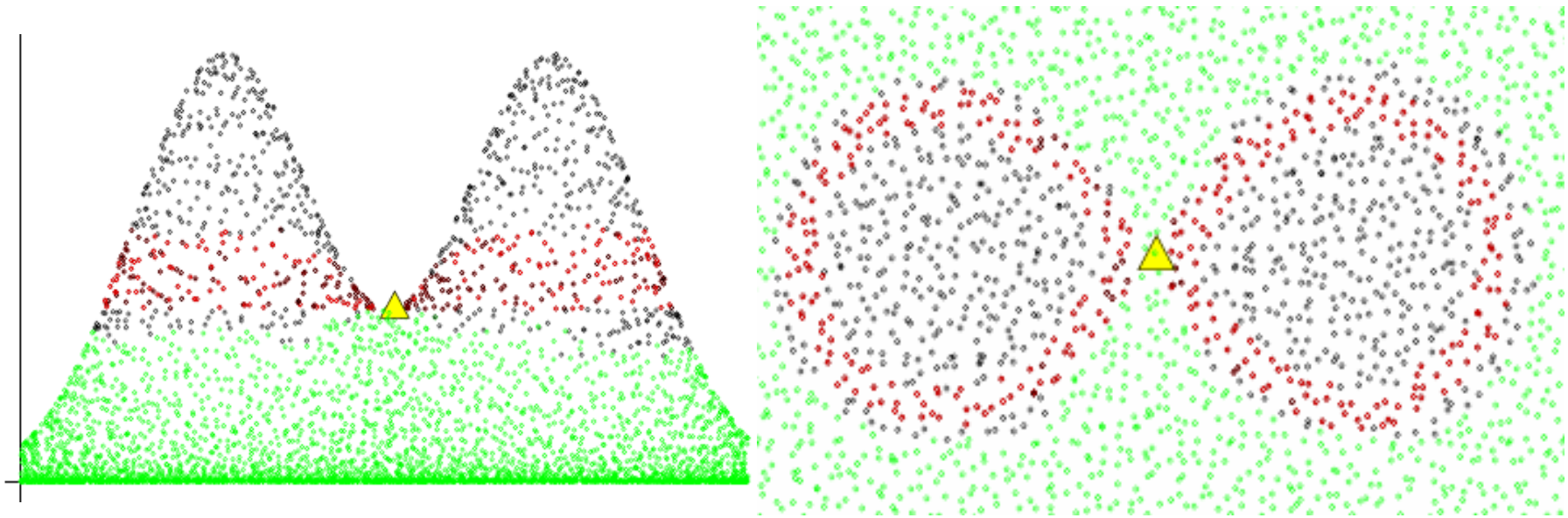
- Start from local maximum.
  - A node can be swept if all its higher neighbors have been swept.
  - The sweep carries the ID of the local MAX.



[Ref : Skraba, Fang, Nguyen, Guibas : IPSN ’06]

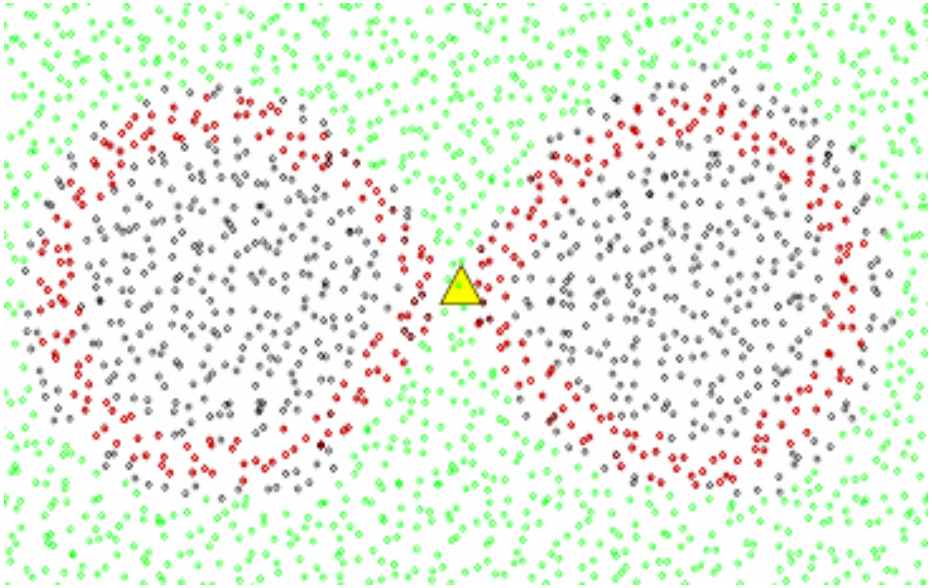
Saddle point

- Saddle: the node with highest value that receives sweep msgs from different critical points.
  - Property: a saddle v is a cut node of the graph on vertices with values above it.



Algorithm to find a saddle

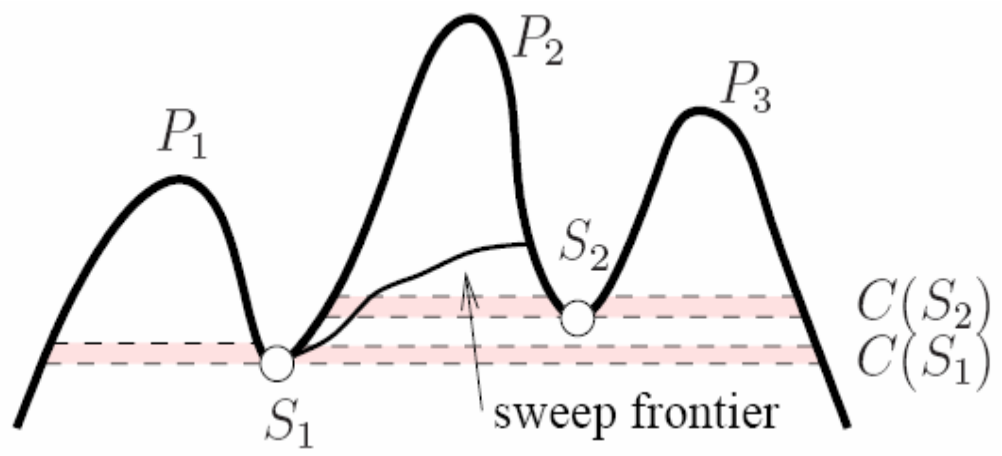
- A node receiving two different sweep msgs
  - Propose to be a saddle candidate
  - Flood the saddle contour (nodes with value above f(v) who has a neighbor below f(v)).



Saddle candidates retire if they meet a candidate with higher value.

Ambiguity?

- Sweep progresses in an asynchronous manner.
  - Contour traversal will verify that all nodes on the saddle contour have been swept properly.



S1 proposes to be saddle for P1, P2 and only wins if all nodes in the contour saddle have been swept by P1 and P2.

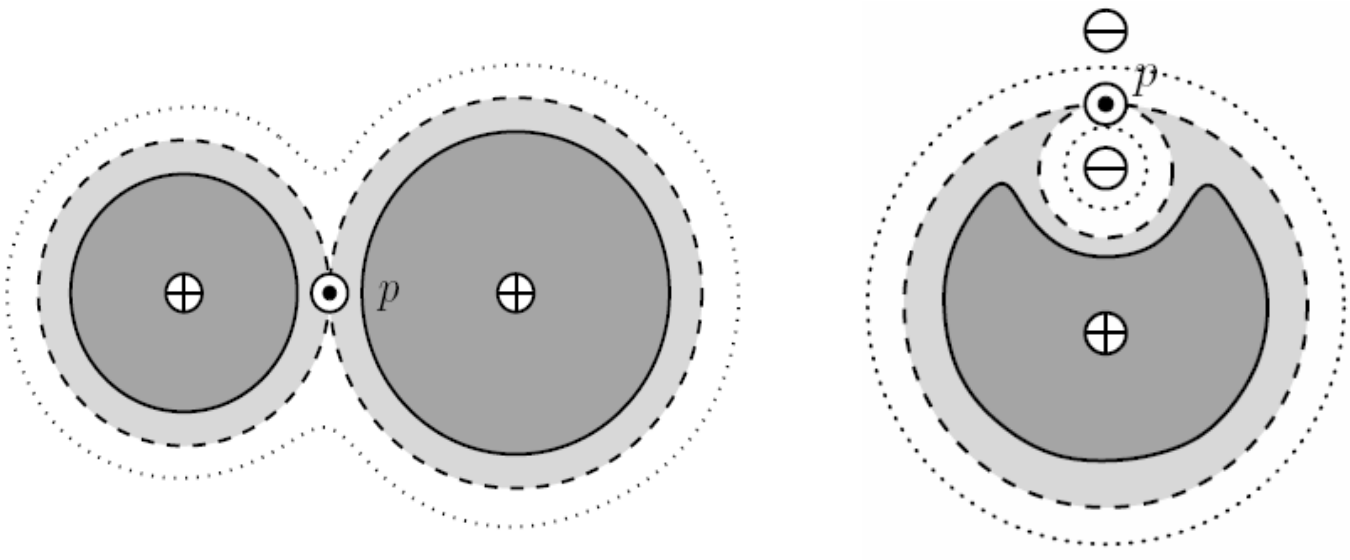
S2 will stop sweep --- this will prevent S1 from winning.

S2 wins and starts a new sweep with ID of S2.

S1 later becomes saddle for S2 and P1

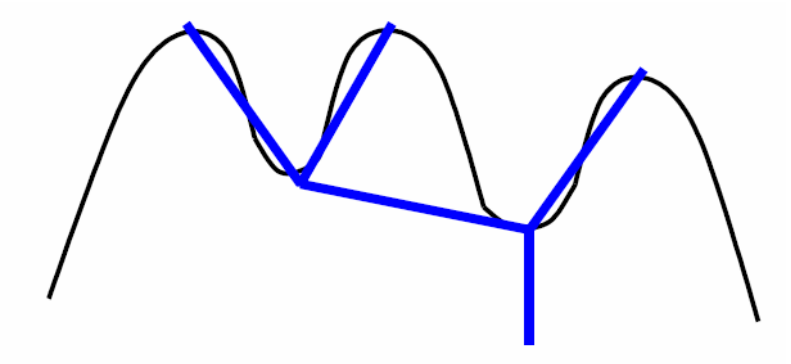
Sweeps

- Top-down sweeps from all local MAX
  - Find all merge saddles.
- Bottom-up sweeps from all local MIN
  - Find all split saddles.

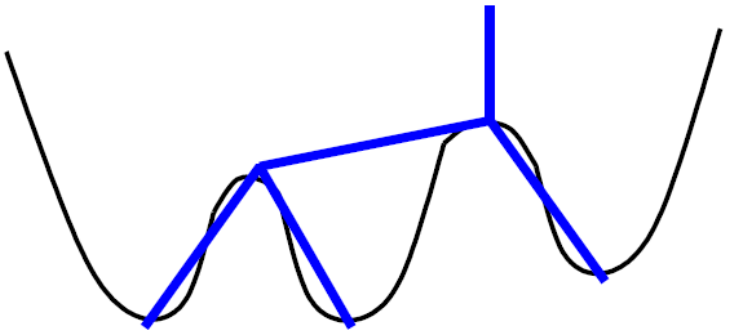


Merge and Split Trees

Merge Tree : Tree on Merge Saddles

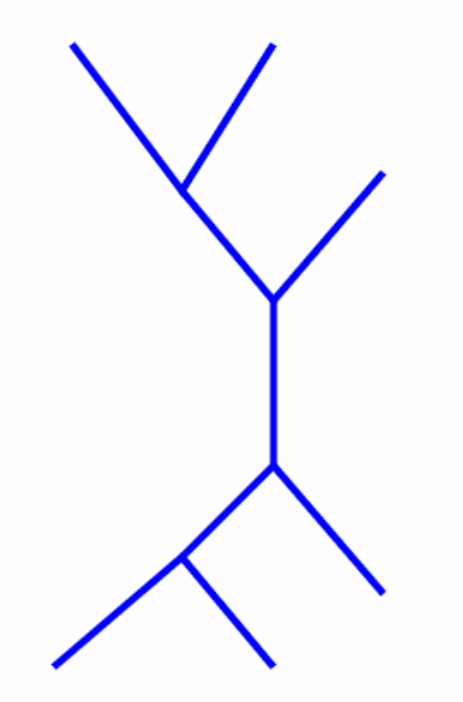


Split Tree : Tree on Split Saddles



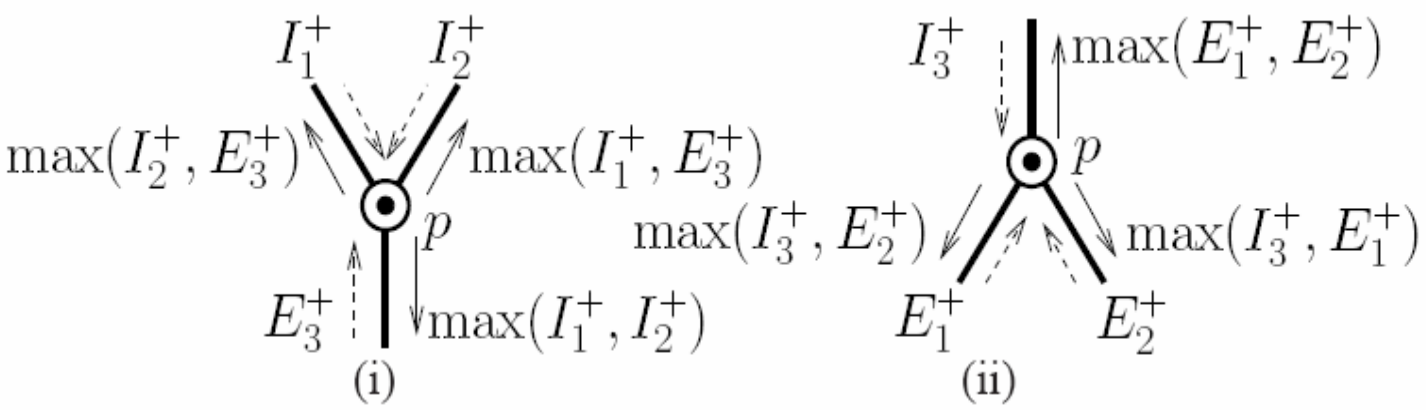
Contour tree

- Merge and split trees combined distributedly
- Get the contour tree



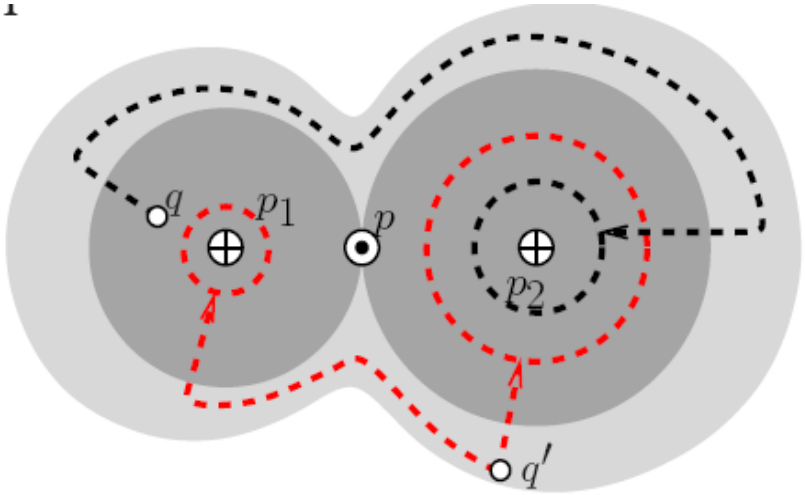
Information Dissemination

- Each internal node of the contour tree knows the range of values reachable through each incident edge.
- Done by simple information dissemination sweeps along the tree.



Gradient Descent to answer Contour Queries

- *"What are the contours at level x ?"*
- At each node *w* of tree,
  - Check if *x* lies in range [min, max] of each edge *e* at *w*
  - If *true,* send message along *e*

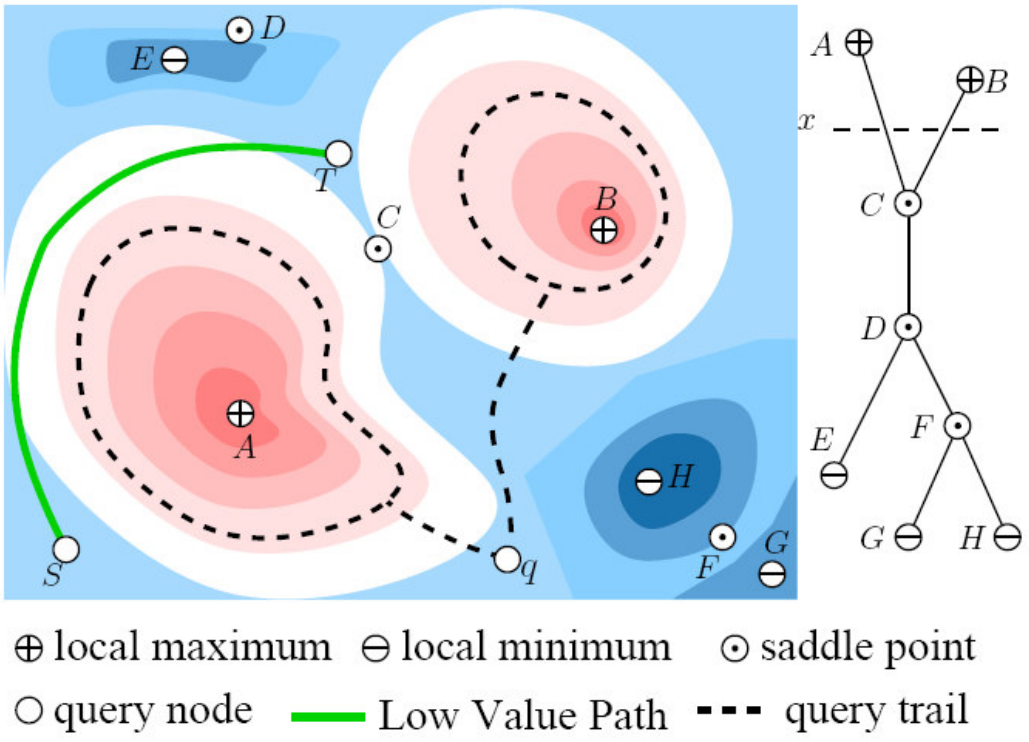


Value restricted routing

- Is there a path between *s* and *t*, that lies entirely in height range *[y,z]* ?

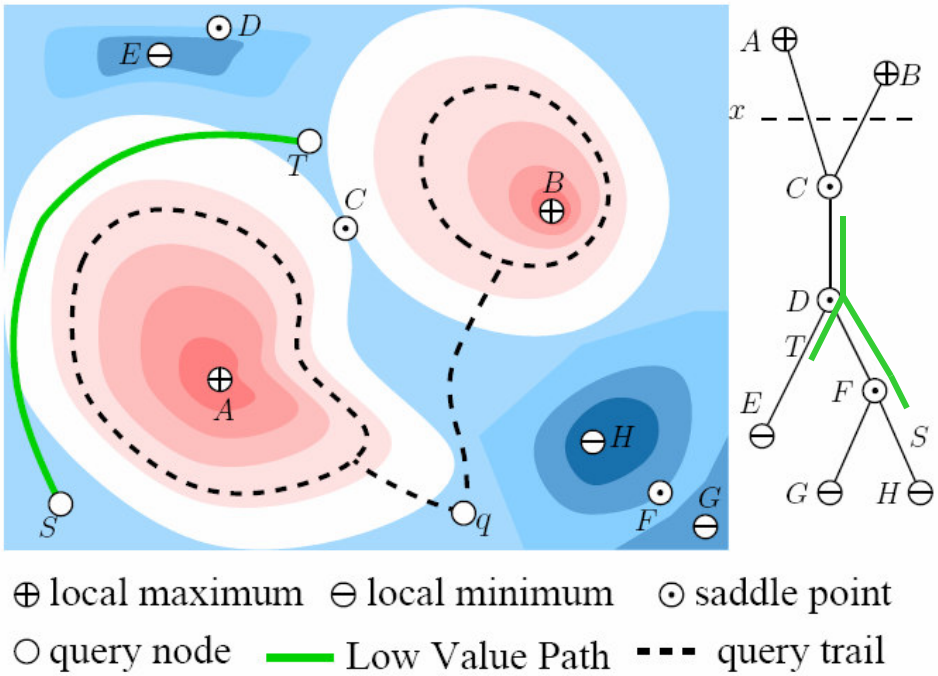
Theorem:

Such a path exists in the network, if and only if a corresponding path exists in the Tree.



Value restricted routing

- Proof Idea :
  - Every connected contour in the domain maps to a single point on a unique edge
  - This map is continuous
  - Tree is a *retract*
    - Points on tree map to themselves
  - It is possible to map any network path to a tree path
  - *Path in tree is unique!*

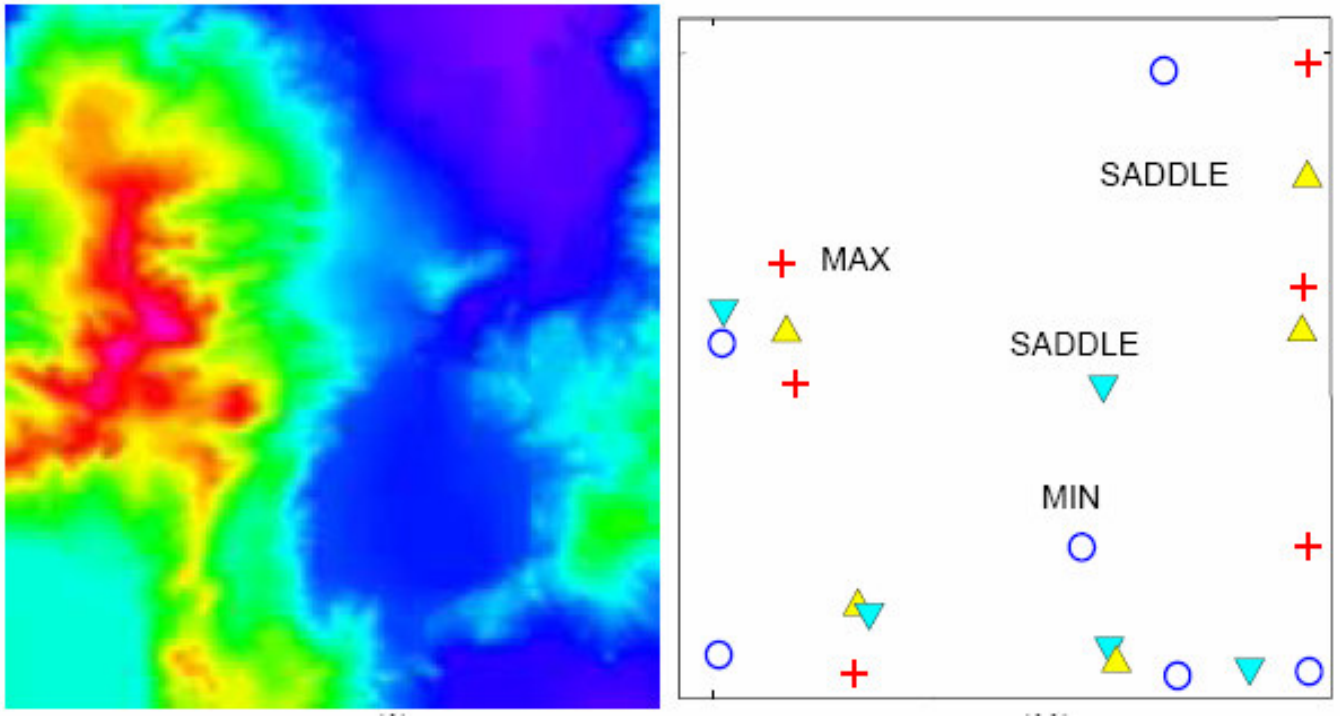


Labeling nodes of the tree

- *log(n)* size label at each node
- Given any 2 nodes, properties of the path in the tree can be derived from their labels
  - check for value restricted paths simply by comparing labels!

Simulations

- Linear preprocessing (tree construction)
- Good load balancing
- Moderate stretch on contour query



Summary

- Distributed contour tree construction
- Gradient descent using the tree finds all components of an Iso-contour.
  - Global routing using greedy local gradient descent
- Value restricted routing : find suitable paths just by comparing labels

Future Work

- Dynamic, time varying contour trees
  - In combination with contour tracking (To be presented later today)
- Efficient noise and plateau handling
- Other applications of contour tree in networks