Continuous Contour Mapping in Sensor Networks

Cheng Zhong and Michael Worboys

National Center for Geographic
Information & Analysis
University of Maine, US

Outline

- Introduction
- Approaches
- CCM Algorithm
- Evaluations
- Conclusions

Introduction

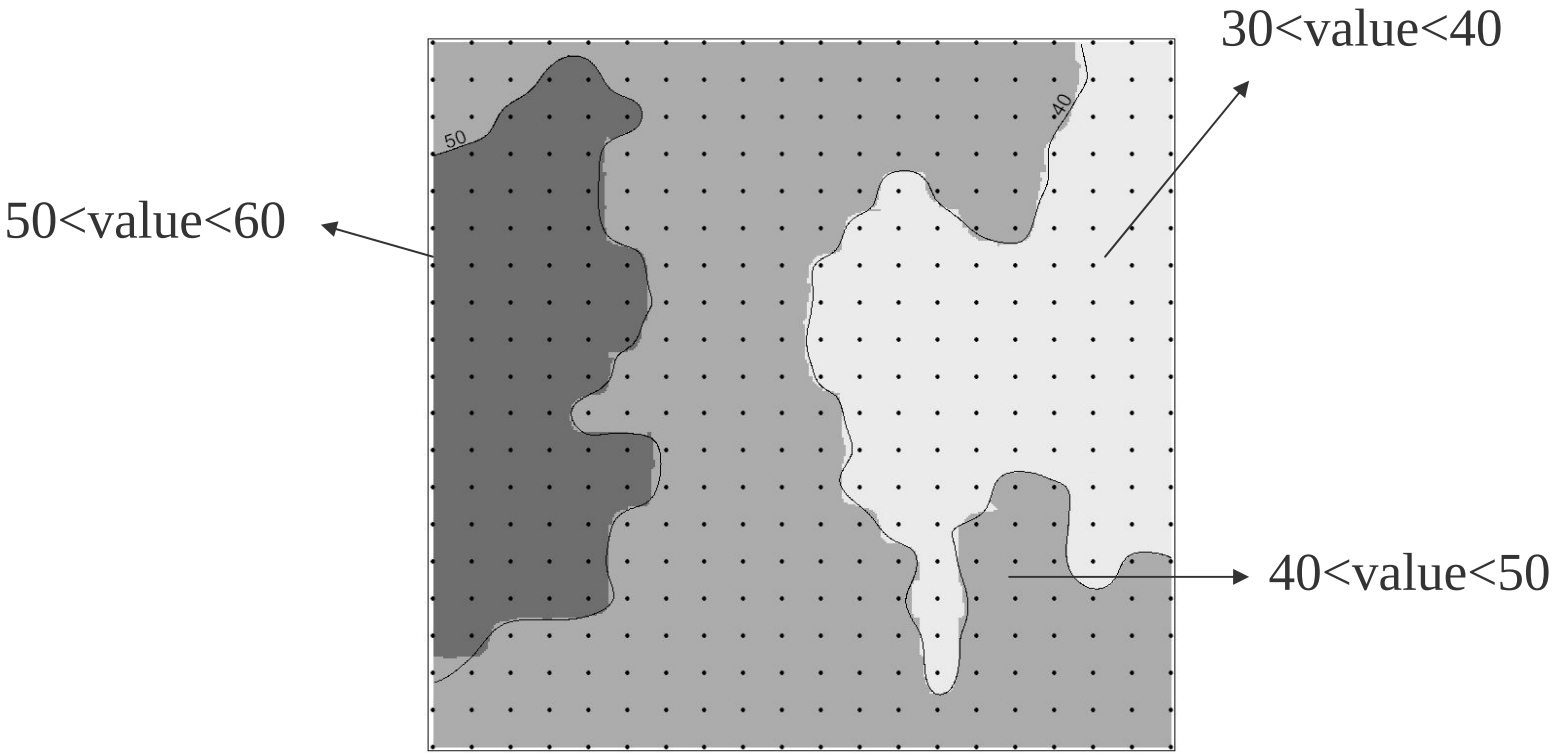
- Motivation
  - What does the whole field look like?
  - Which regions have values between 50 and 60?
  - Try to answer these questions using contour maps

Introduction

- Motivation
  - A contour map is a useful data representation schema that provides an efficient way to visualize the field monitored by sensor networks
  - Generate contour maps for the region
  - Reduce data transmission

Introduction

- Contour map example



Outline

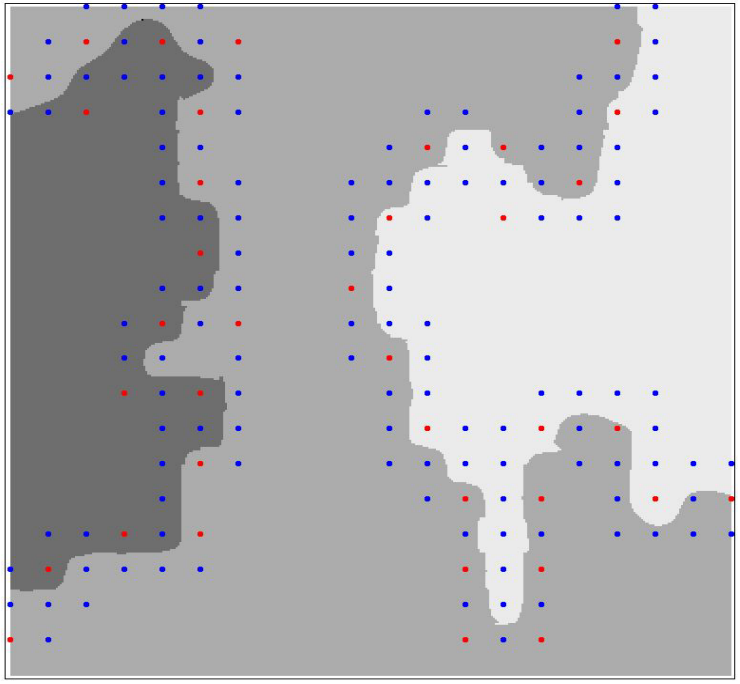
- Introduction
- Approaches
- CCM Algorithm
- Evaluations
- Conclusions

Approaches

- Contour detection is based on node-neighbor communication. If a node and one of it's neighbors are in different values ranges. At least, there is one contour between them

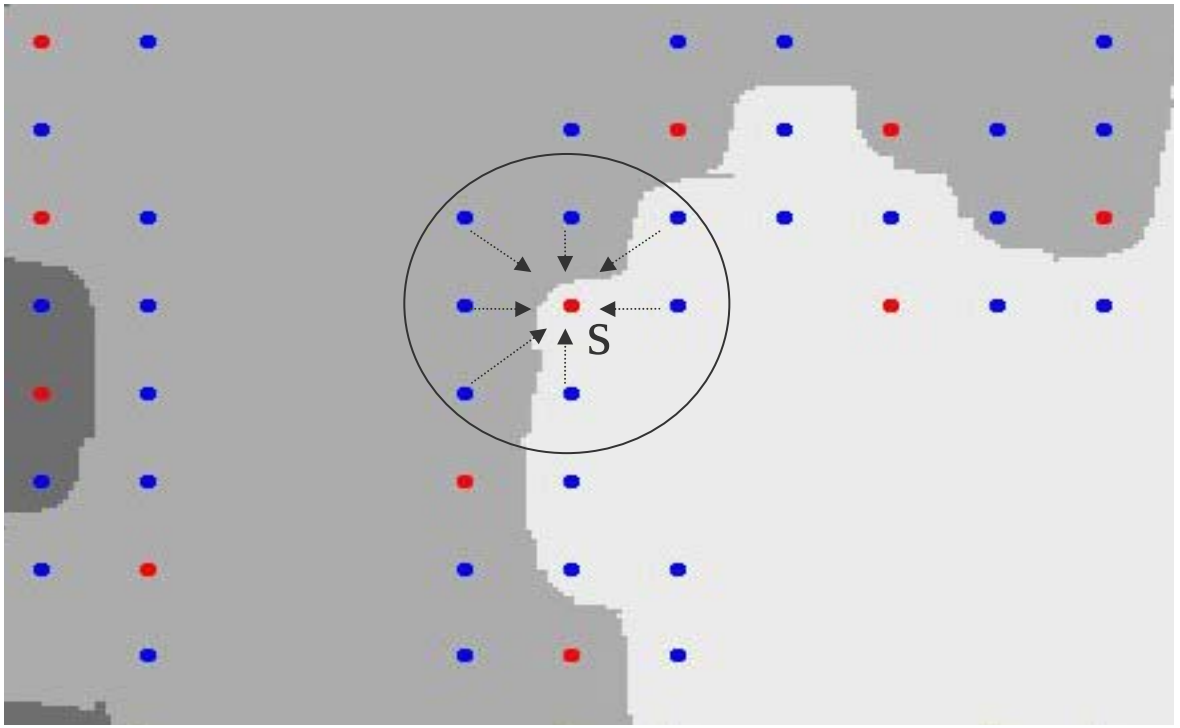
Approaches

- Design an algorithm (CCM) for reporting node (in red) selection



Approaches

- Design a data structure for neighbor suppression

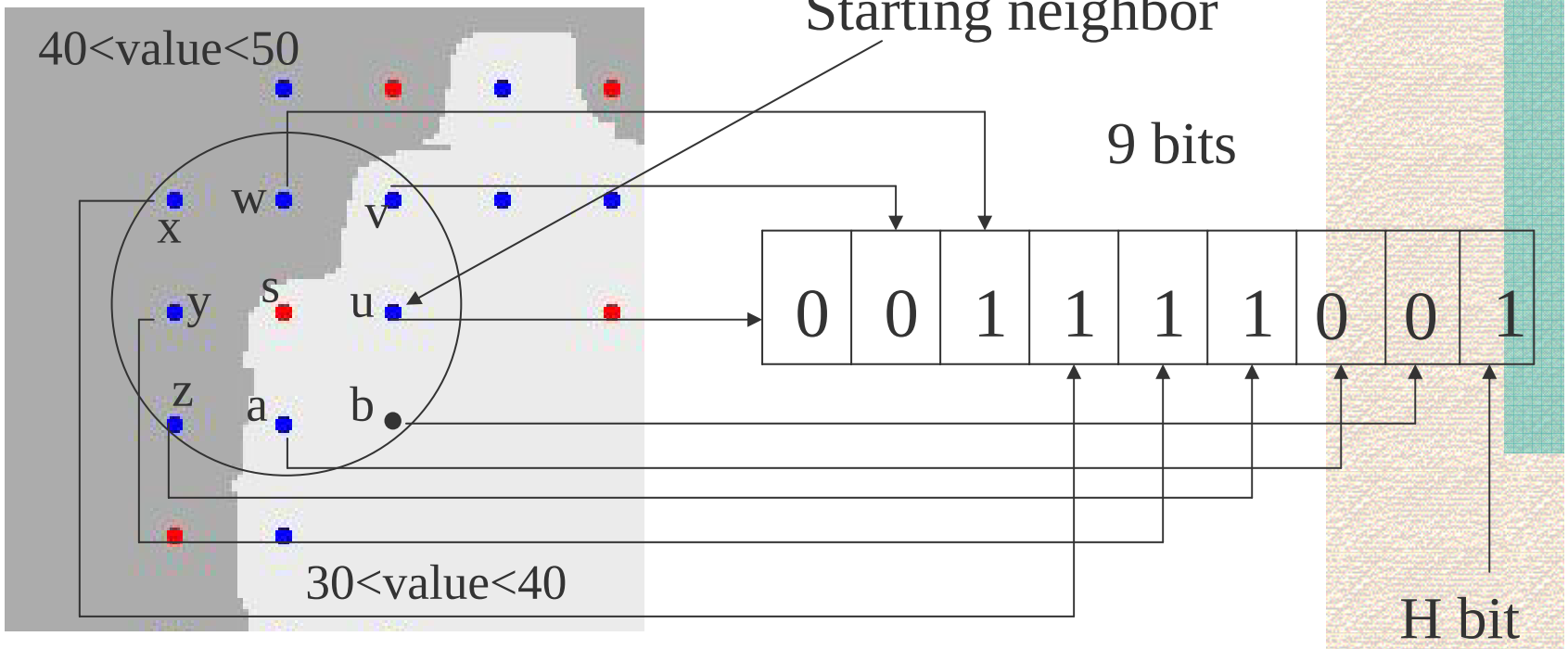


Approaches

- Contour neighbor array (CN-array)
  - Each node sets a starting neighbor. The CN-array of the node corresponds to its neighbors by counterclockwise order with an additional h bit.
- Each reporting node sends its ID, reading and CN-array back

Approaches

- CN-array example



By neighbor ID & reading pairs:
(2 bytes + 2 bytes) * 4 = 16 bytes

Outline

- Introduction
- Approaches
- CCM Algorithm
- Evaluations
- Conclusions

CCM Algorithm

- Reporting node selection
  - **Step 1:** u checks if it is a contour node. If not, u skips the following steps
  - **Step 2:** If u is a reporting node in the previous round, and u has unchanged CN-array and value range, u sends its ID back and suppresses its neighbors and skips step 3
  - **Step 3:** If u is not suppressed after a random time, u becomes a reporting node. u reports its ID, reading and CN-array to the sink and suppresses its neighbors

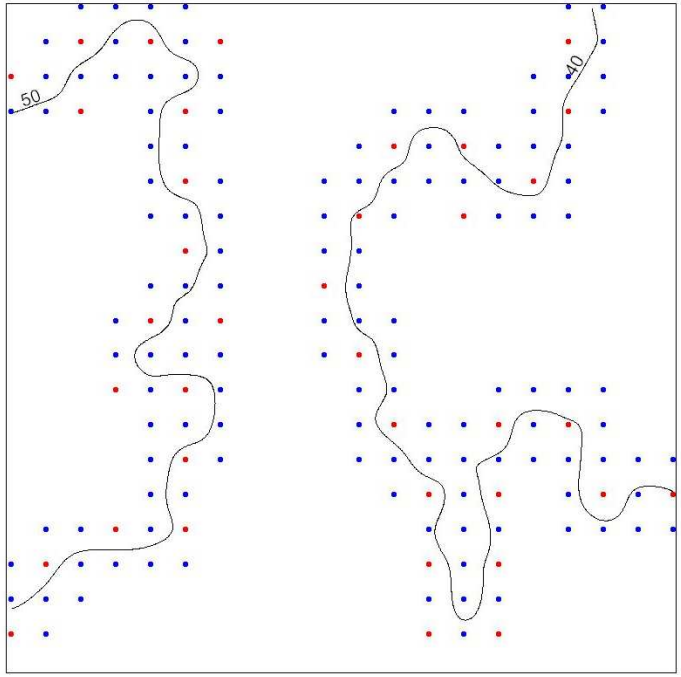
CCM Algorithm

- Optimization by a greedy approach: let contour nodes with large contour neighbor count report and suppress their neighbors first

CCM Algorithm

- Example output

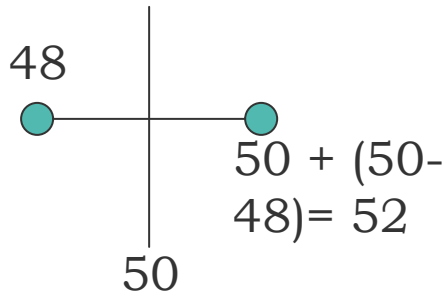
  There are 165 contour nodes (in blue and red) that can detect contours. Only 37 nodes (in red) will report to the sink



CCM algorithm

- The base station interpolates values to non-reporting contour nodes using CN-array and the neighbor ordering information which is sent by each node at the network initialization phase (mapping CN-array to neighbors)

$$R(u) = \frac{\sum_{s_i \in S_{u1}} R(s_i) + \sum_{s_j \in S_{u2}} 2 * C(u, s_j) - R(s_j)}{|S_{u1}| + |S_{u2}|}$$



Outline

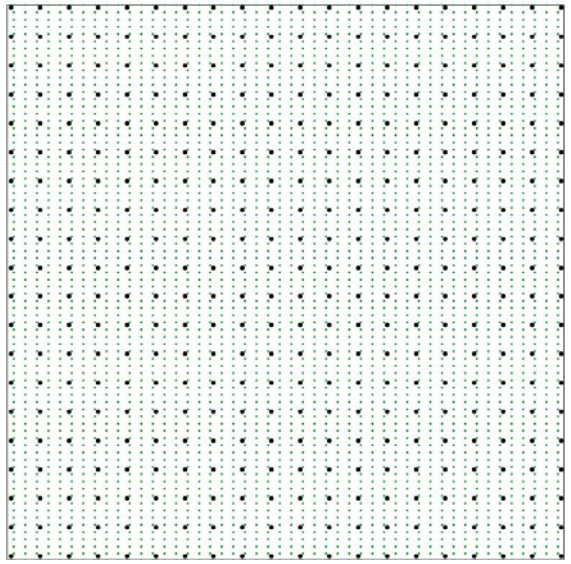
- Introduction
- Approaches
- CCM Algorithm
- Evaluations
- Conclusions

Evaluations

- Simulation setup
  - A 400m * 400m square field. 400 nodes are evenly deployed and the node communication radius is 30m. 2 Bytes is used for CN-array representation
- Baseline algorithms
  - Spatial suppression algorithm
  - Temporal suppression algorithm
  - Isolines algorithm

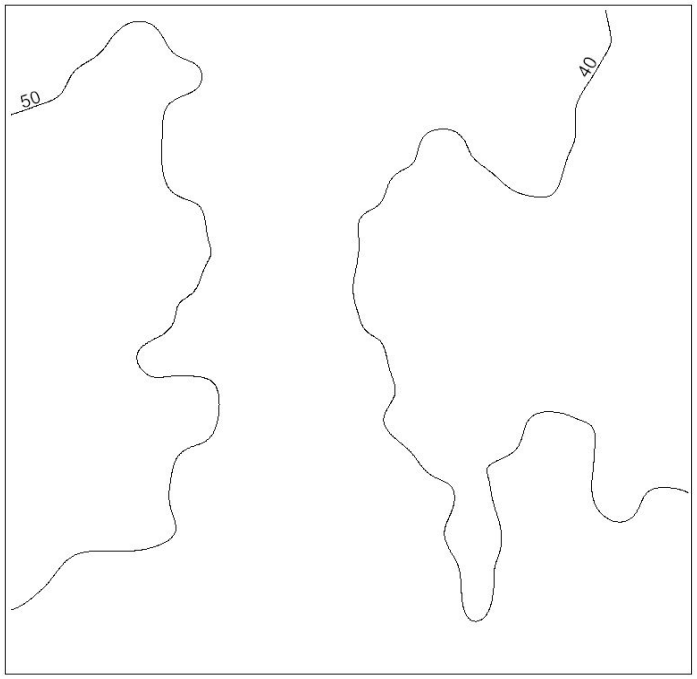
Evaluations

- Map Similarity
  - interpolate 4000 points. generate contours using ArcGIS View tool and calculate the percentage of points that are actually in the correct value ranges.

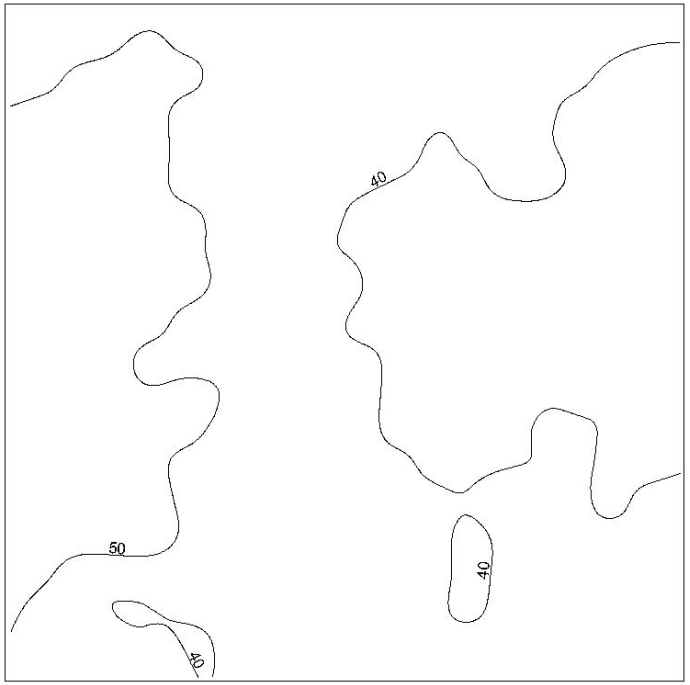


Evaluations

- The baseline map VS. a generated map



baseline map



map using CCM data

Evaluations

- Similarity and data transmission for snapshot detection

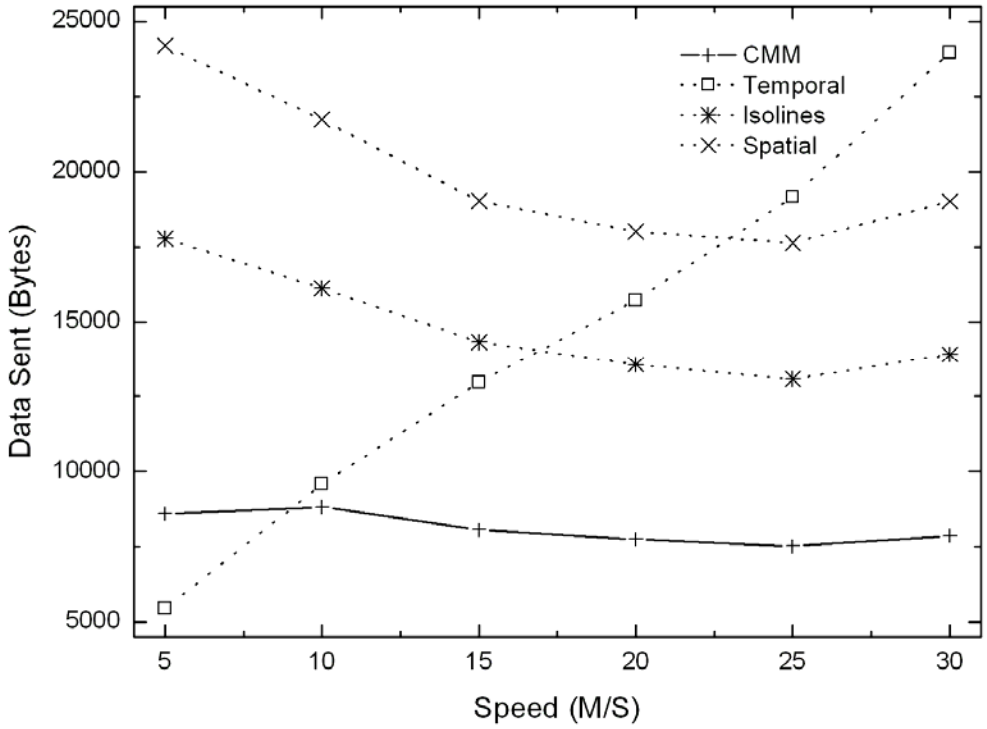
	Similarity	Data sent (Bytes)
Spatial	93.1% (sd 2%)	11008 (sd 379)
Isolines	92.58% (sd 1.93%)	8128 (sd 327)
CCM*	95.39% (sd 0.69%)	5336 (sd 322)
CCM	94.59% (sd 0.93%)	4527 (sd 169)

Evaluations

- Continuous case
  - A front moves from left to right at different speeds
  - Continuously sample 6 times from the first second

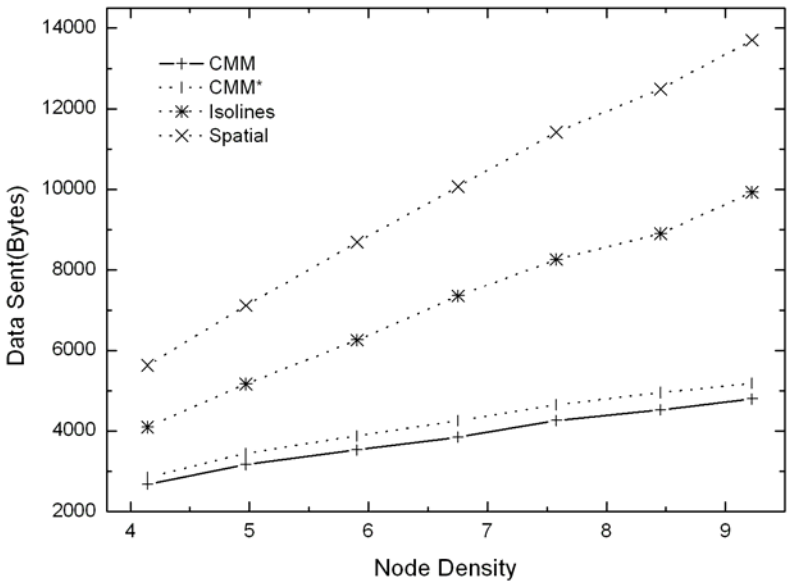
Evaluations

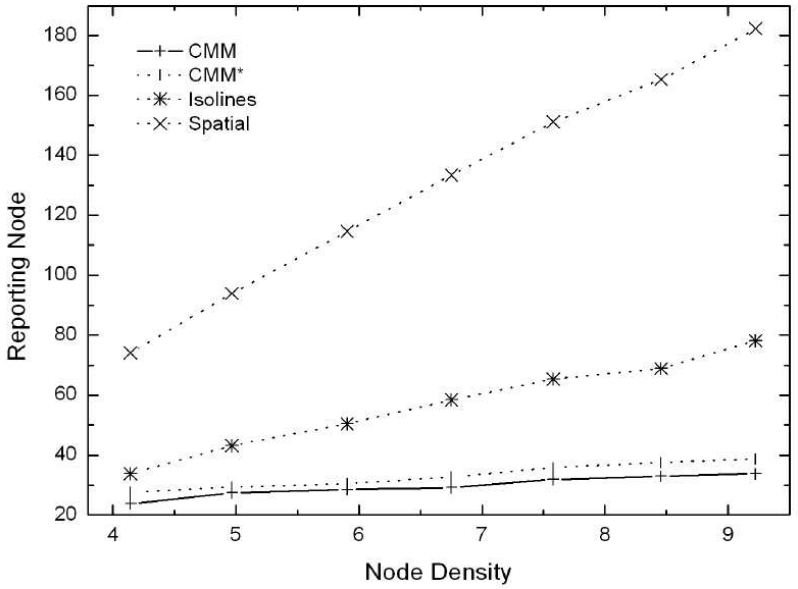
- Continuous case



Evaluations

- The impact of node densities





Outline

- Introduction
- Approaches
- CCM Algorithm
- Evaluations
- Conclusions

Conclusions

- Conclusions
  - Use a bit array to represent neighborhood information for each reporting node
  - Use a random timer algorithm to select reporting nodes
  - Out-performs baseline algorithms and is more scalable

Conclusions

- Existing problems
  - If a reporting node and its neighbors are not in the adjacent value ranges, the sink cannot interpolate good values to nodes
  - Can not deal with node failure well

Thanks

- This work is supported by the US National Science Foundation under grant number II-0504494

Thanks

- Questions?