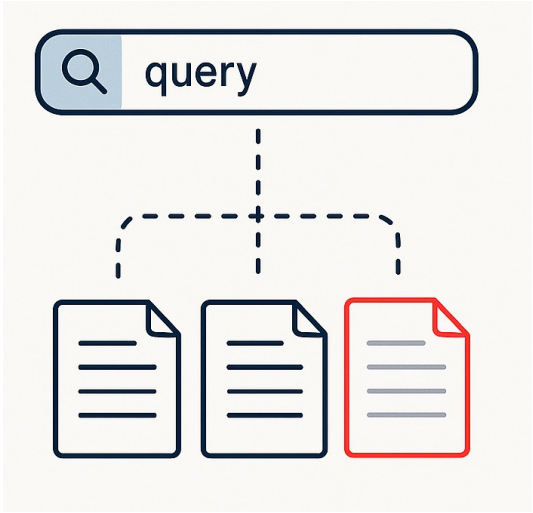
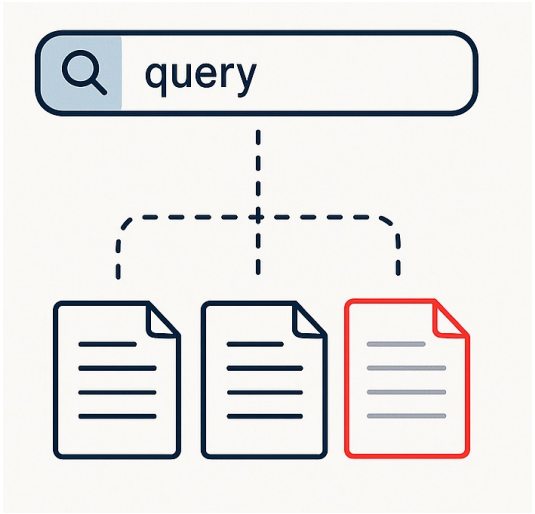
Graph-based Nearest Neighbors with Dynamic Updates via Random Walks

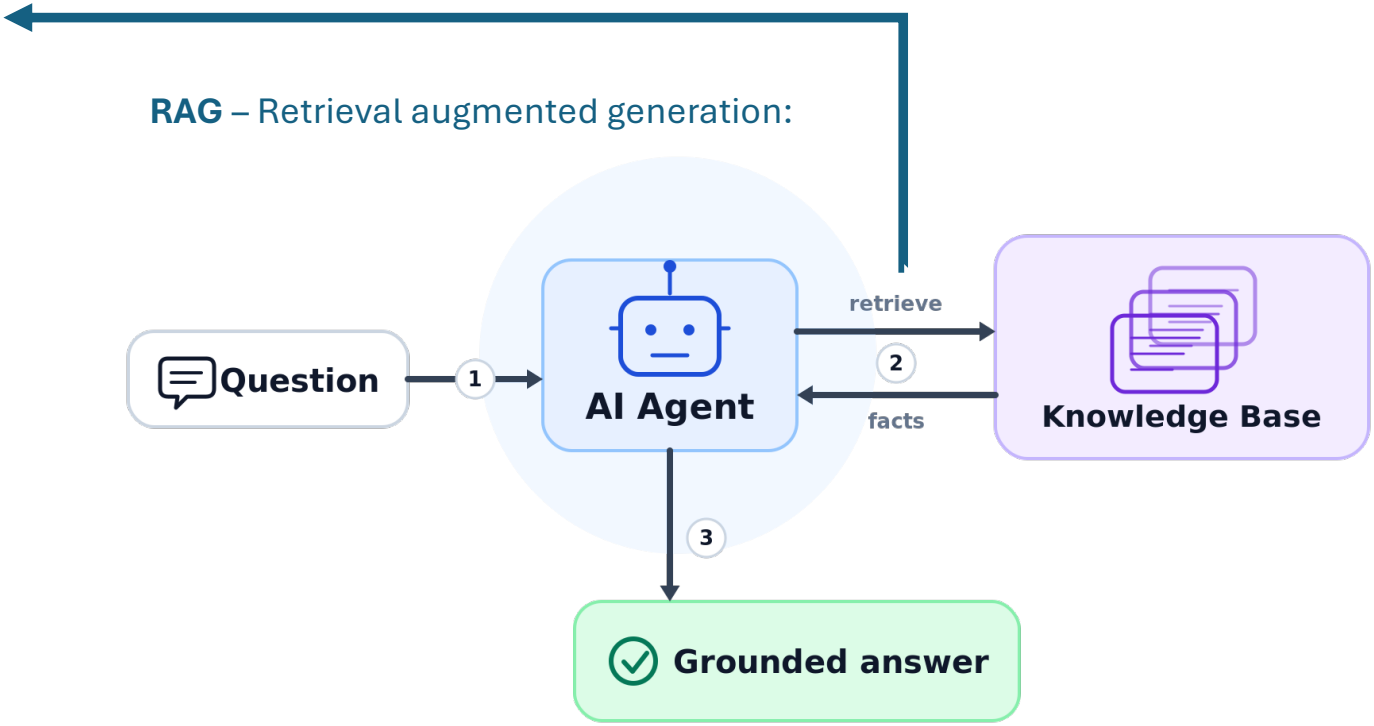
Nina Mishra, Yonatan Naamad, Tal Wagner, Lichen Zhang

Approximate Nearest Neighbor Search (ANN)

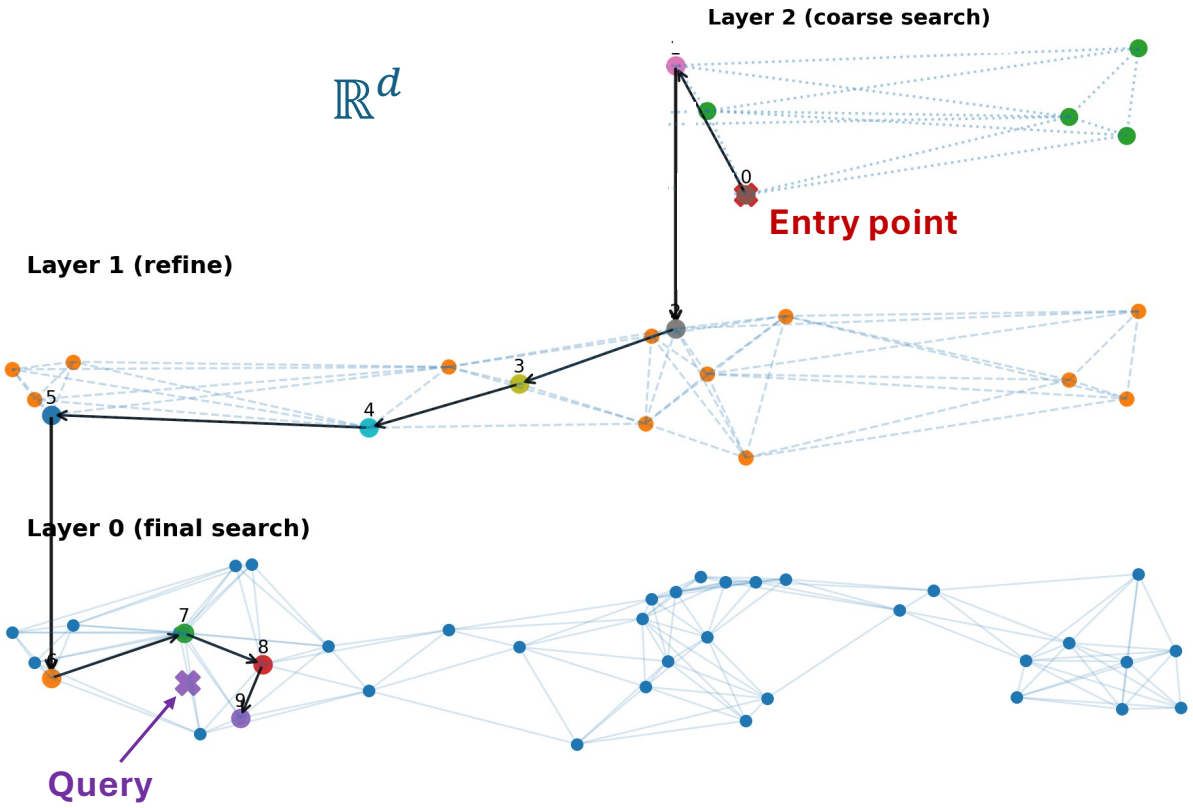


Approximate Nearest Neighbor Search (ANN)

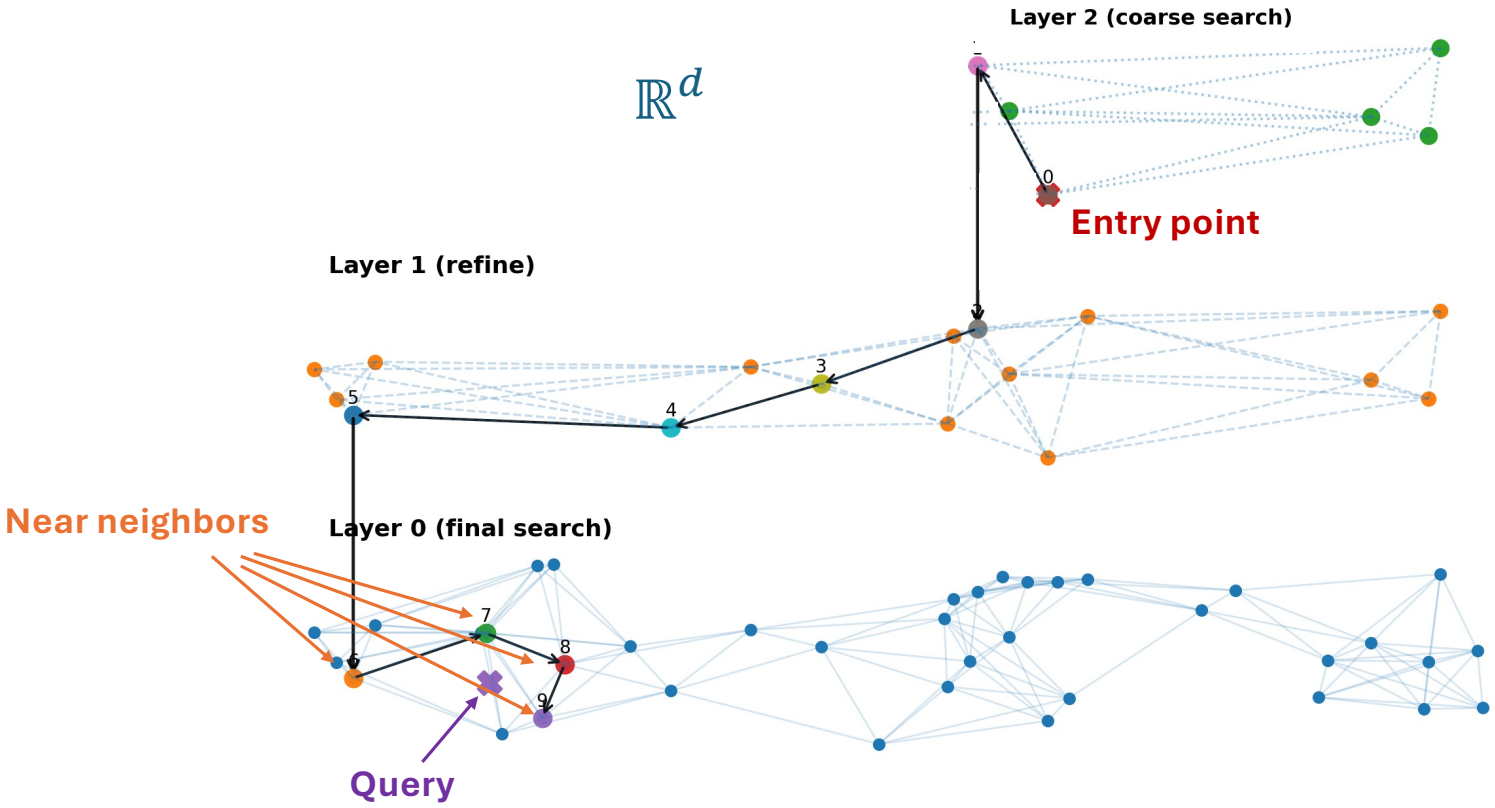




Graph Search Methods in ANN



Graph Search Methods in ANN



ANN with Dynamic Updates

- **Challenge:** Data is **dynamic**

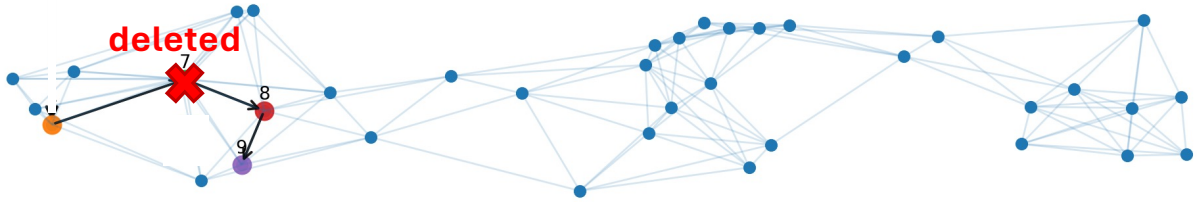
ANN with Dynamic Updates

- **Challenge:** Data is **dynamic**

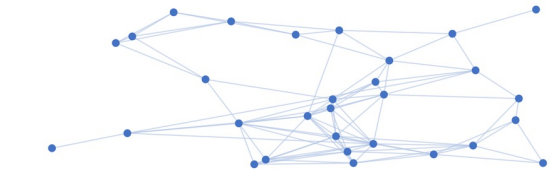
ANN with Dynamic Updates

- **Challenge:** Data is **dynamic**

- **How to update the navigation graph?**
  - Insertions are easier
  - **Deletions are harder**

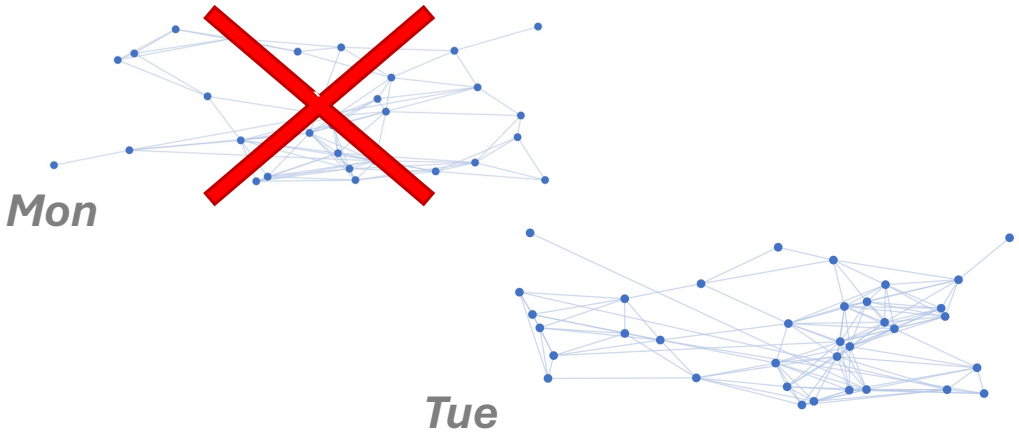


Vanilla Solution 1: Periodic Rebuilds

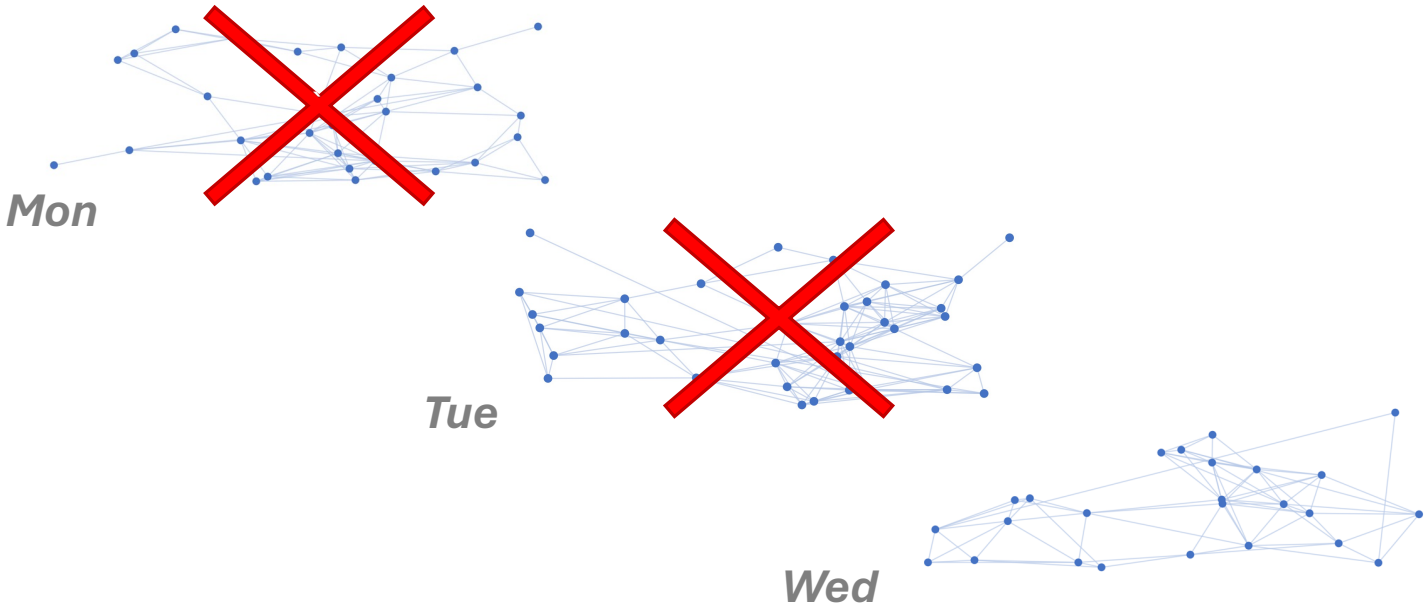


Mon

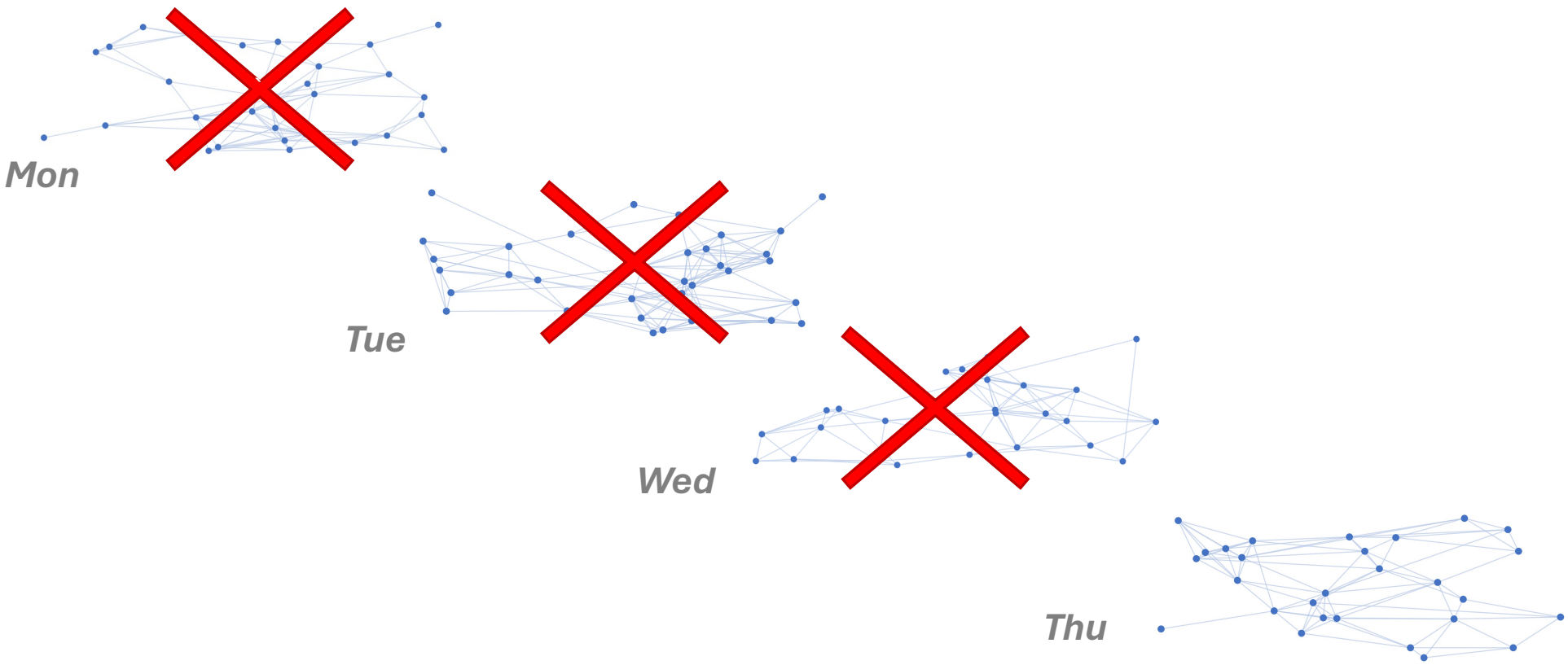
Vanilla Solution 1: Periodic Rebuilds



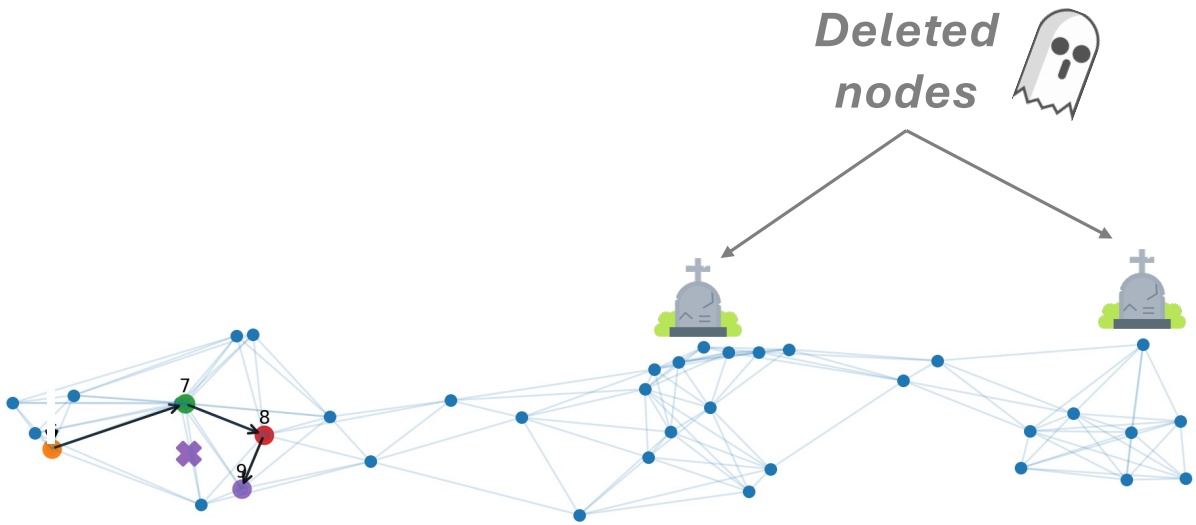
Vanilla Solution 1: Periodic Rebuilds



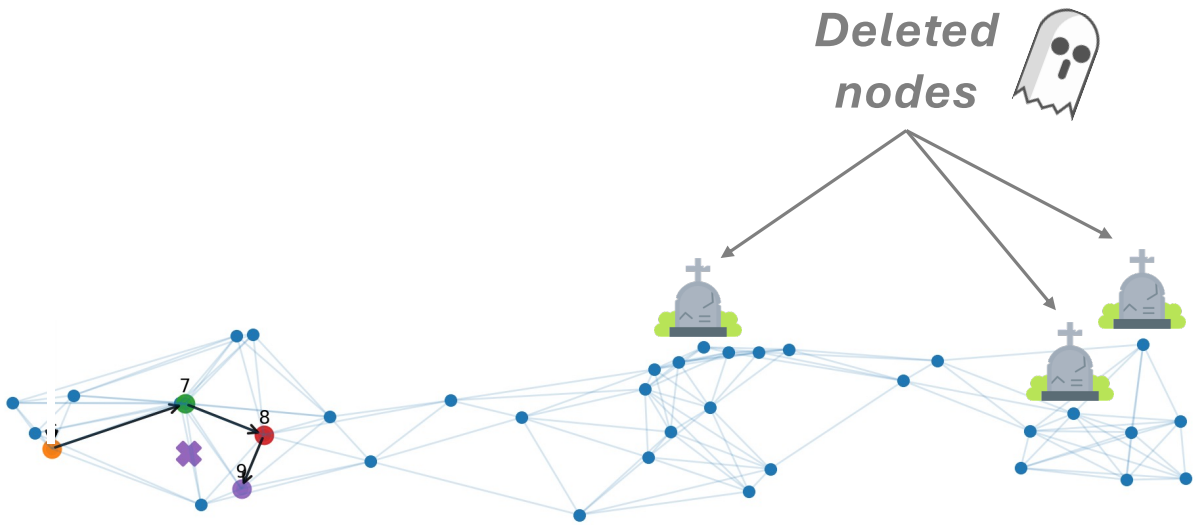
Vanilla Solution 1: Periodic Rebuilds



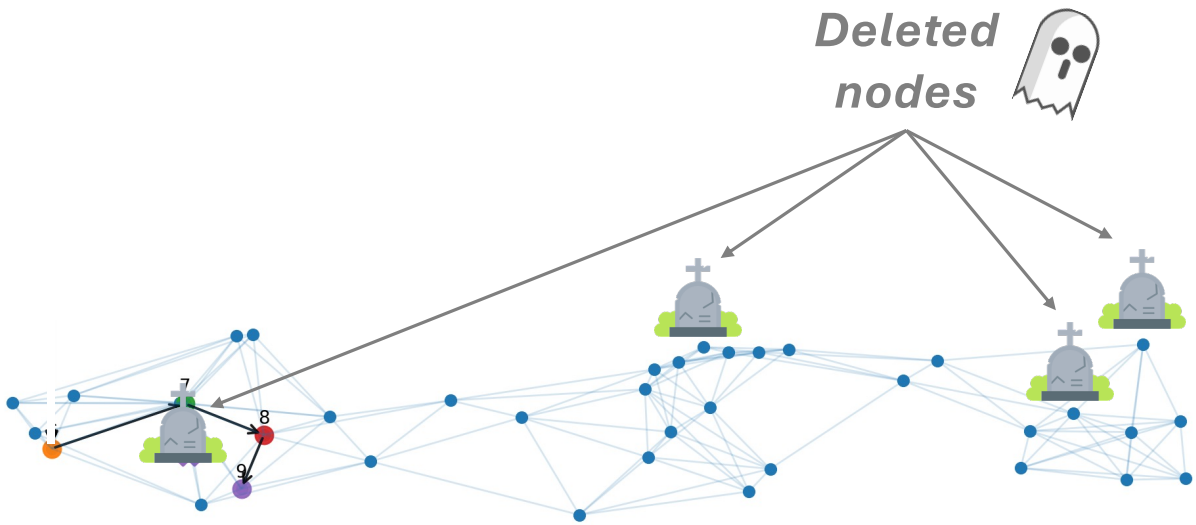
Vanilla Solution 2: Tombstones



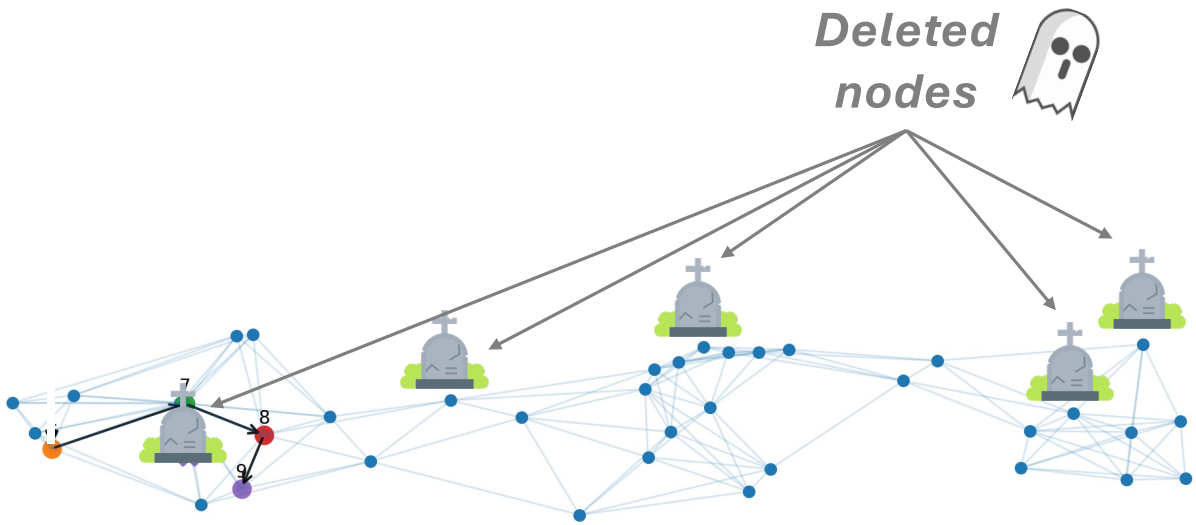
Vanilla Solution 2: Tombstones



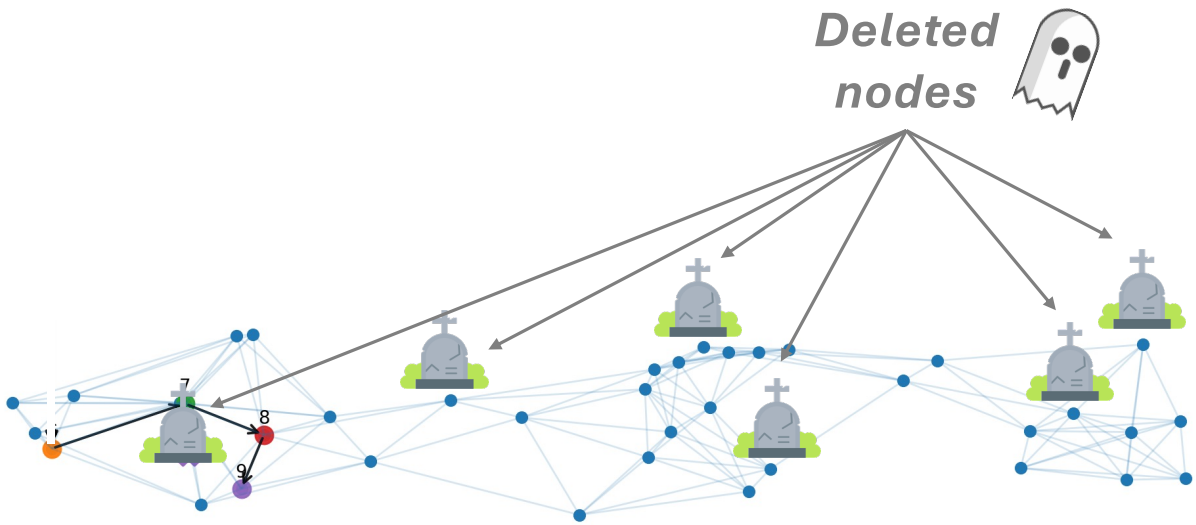
Vanilla Solution 2: Tombstones



Vanilla Solution 2: Tombstones



Vanilla Solution 2: Tombstones



Our Approach: Local Patching

- **But how?**

Our Approach: Local Patching

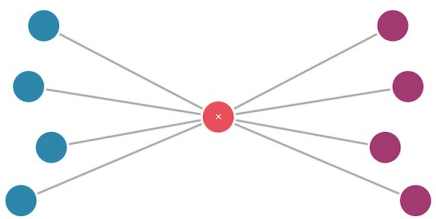
- **But how?**

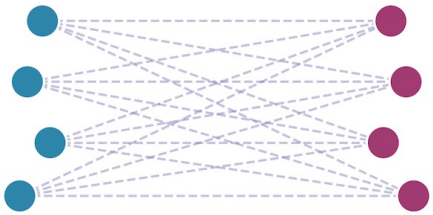
- Our method: ***SPatch* – Sparsified Patching**

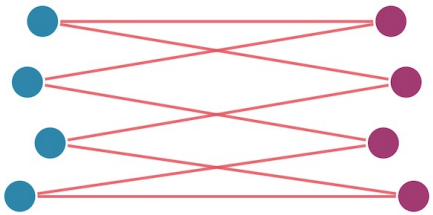
Our Approach: Local Patching

- **But how?**

- Our method: ***SPatch* – Sparsified Patching**



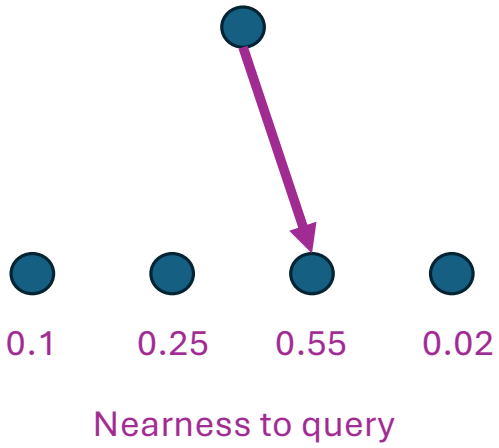




Our formulation: “Softmax Walk”

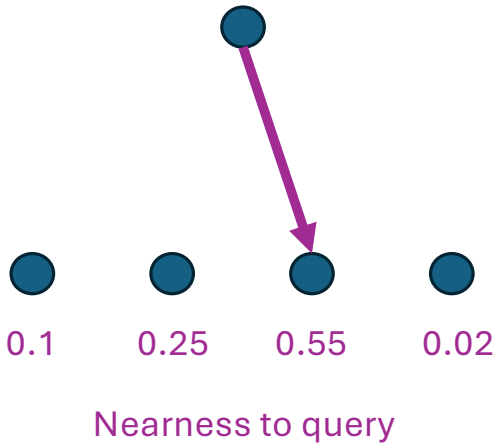
Our formulation: “Softmax Walk”

Greedy walk (vanilla graph-search):
Deterministic walk

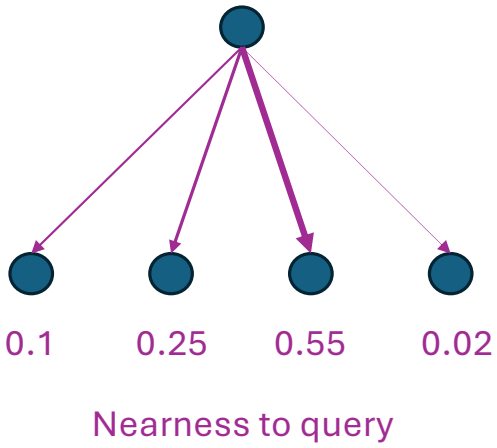


Our formulation: “Softmax Walk”

Greedy walk (vanilla graph-search):
Deterministic walk



Softmax walk (our conceptual analog):
Random walk

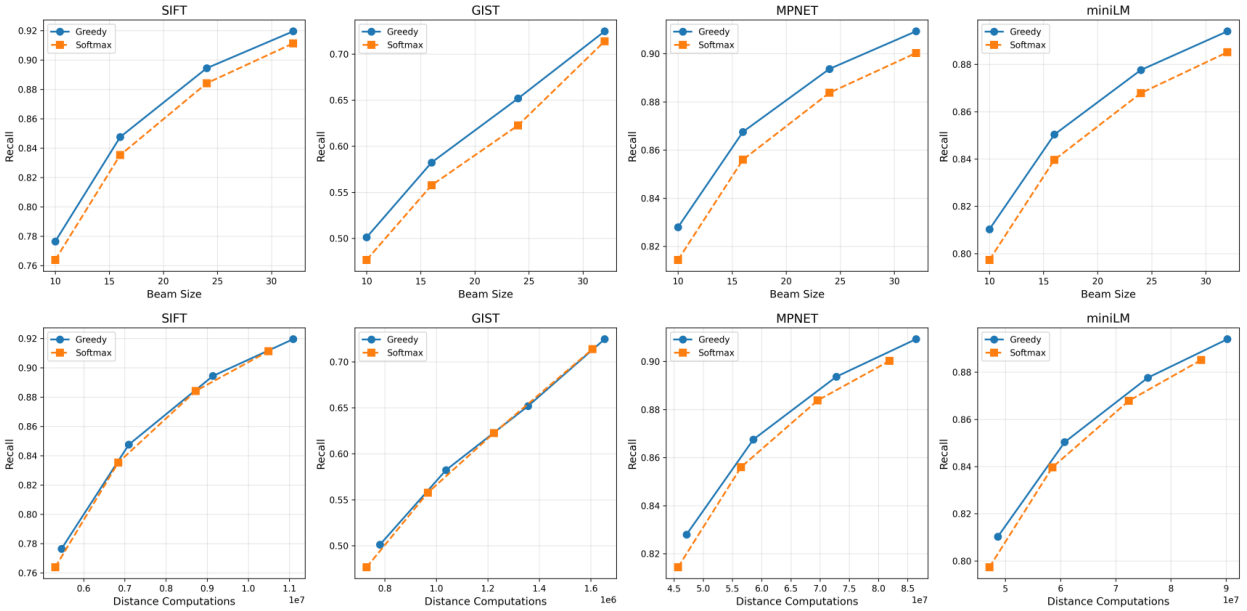


Softmax Walk: Why?

- Access to random walk theory on graphs
- Leads to theoretically motivated patching algorithm (SPatch)

Softmax Walk: Why?

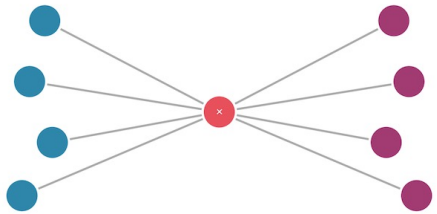
- Access to random walk theory on graphs
- Leads to theoretically motivated patching algorithm (SPatch)
- Maintains empirical performance:



SPatch

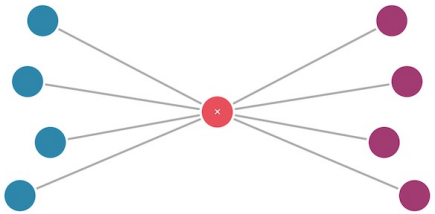
SPatch

1. Delete node

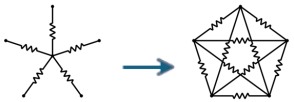


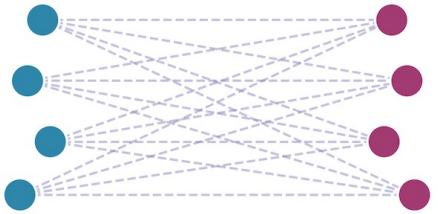
SPatch

1. Delete node



2. Star-mesh transform

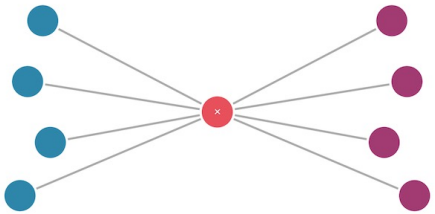


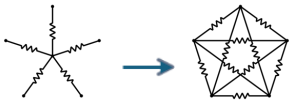


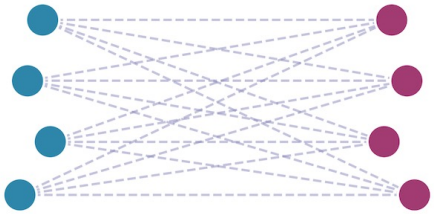
- Restores random walk connectivity
- Makes neighborhood too dense

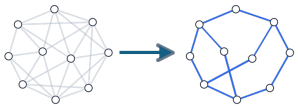
SPatch

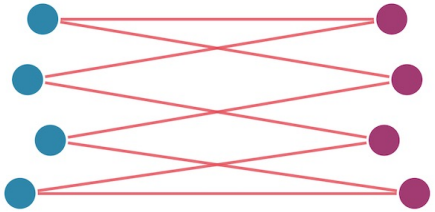
1. Delete node



2. Star-mesh transform 
   - ➢ Restores random walk connectivity
   - ➢ Makes neighborhood too dense

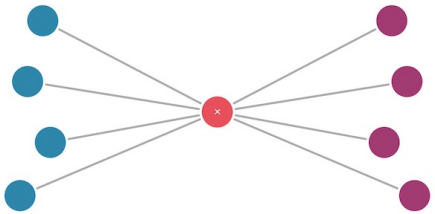


3. Spectral sparsification 
   - ➢ Approximately preserves restored connectivity
   - ➢ Keeps neighborhood sparse

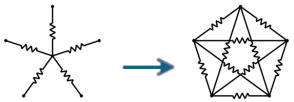


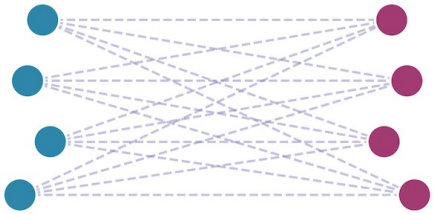
SPatch

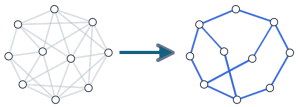
1. Delete node

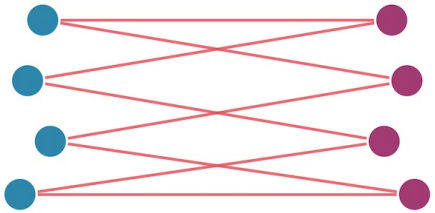


Spectral graph theory

2. Star-mesh transform 
   - ➢ Restores random walk connectivity
   - ➢ Makes neighborhood too dense



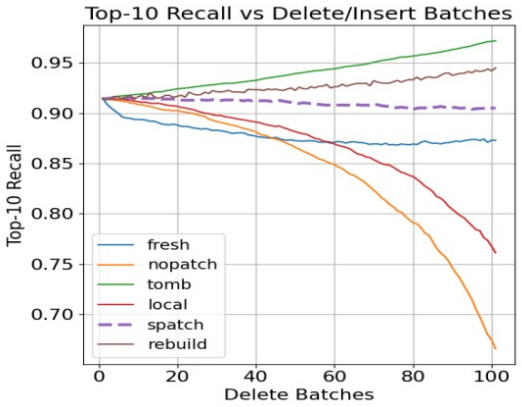
3. Spectral sparsification 
   - ➢ Approximately preserves restored connectivity
   - ➢ Keeps neighborhood sparse

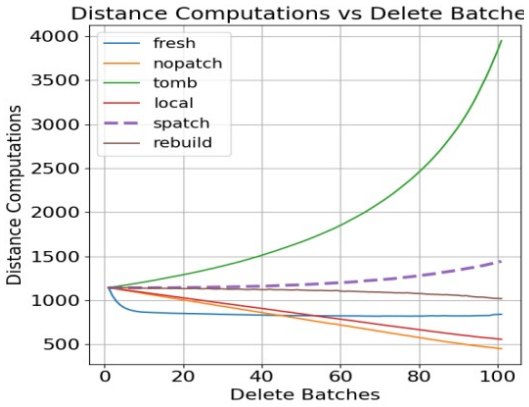


Experiments

Method	Recall	Speed	Del Time	Space
Tombstone	✓	×	✓	×
No patch	×	✓	✓	✓
Local	×	✓	✓	✓
FreshDiskANN	✓	✓	×	✓
Global	✓	✓	×	✓
`SPatch` (ours)	✓	✓	✓	✓

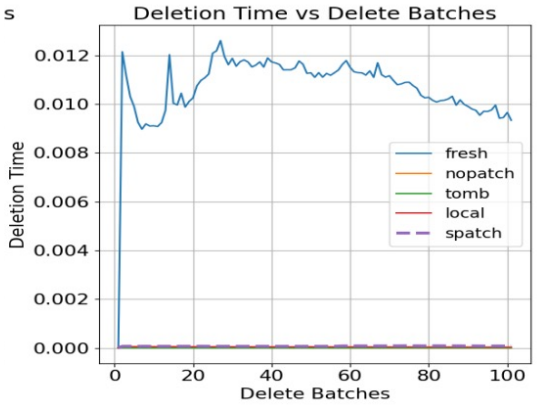
Accuracy

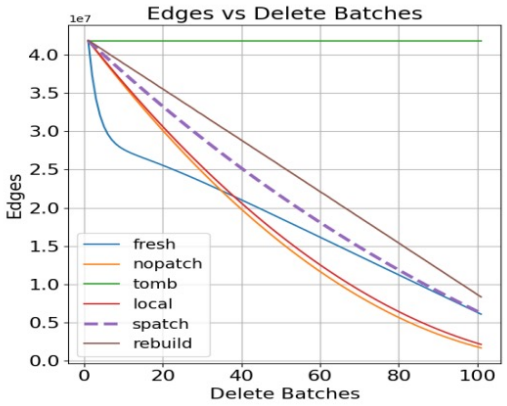




Query time

Deletion time





Memory usage

Conclusion

- **SPatch** is a new deletion algorithm for graph-based nearest neighbor search indices like HNSW
  - Mathematically grounded in spectral graph theory
  - Based on a “softmax walk” formulation
  - Efficient and accurate empirically

- **Thank you**