Universität Konstanz

Locality Optimization for traversal-based Queries on Graph Databases

Fabian Klopfer

Databases and Information Systems Group
Department of Computer and Information Science
University of Konstanz, 30.04.2021

Outline

1 Introduction

2 Background

3 Problem Definition

4 Locality-optimizing Record Layout

5 Evaluation

6 Conclusion

Motivation I

Current state of performance-optimized

- relational databases:
  - Storage order determined by join attribute. $\rightarrow$ enables localized access.
  - Explicit clustered indices (often B+-Trees). $\rightarrow$ represents order, speeds up range queries

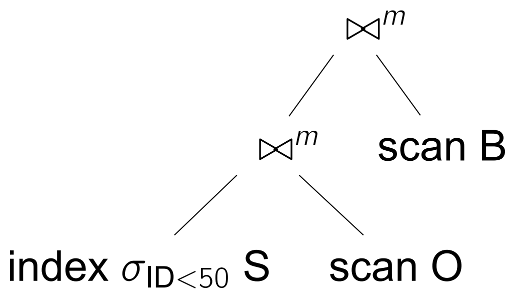
  $\Rightarrow$ Accesses are made as sequential as possible.

- graph databases:
  - Storage order determined by insertion order.
  - Implicit, possibly unclustered index for relationships (doubly-linked list).
  - Lucene-based indexes on properties, labels, . . . , unclustered.

  $\Rightarrow$ Access is mostly random.

Example I

Show me all boats owned by sailors with an ID less than 50:



Reads are mostly sequential.
$\Rightarrow$ Prefetch & cache hit.

Example II

Nodes are Sailors and Boats, relationships “owns”

$$\sigma_{y.\text{label} = \text{Boat}}$$

$$\uparrow_{x}^{y}$$

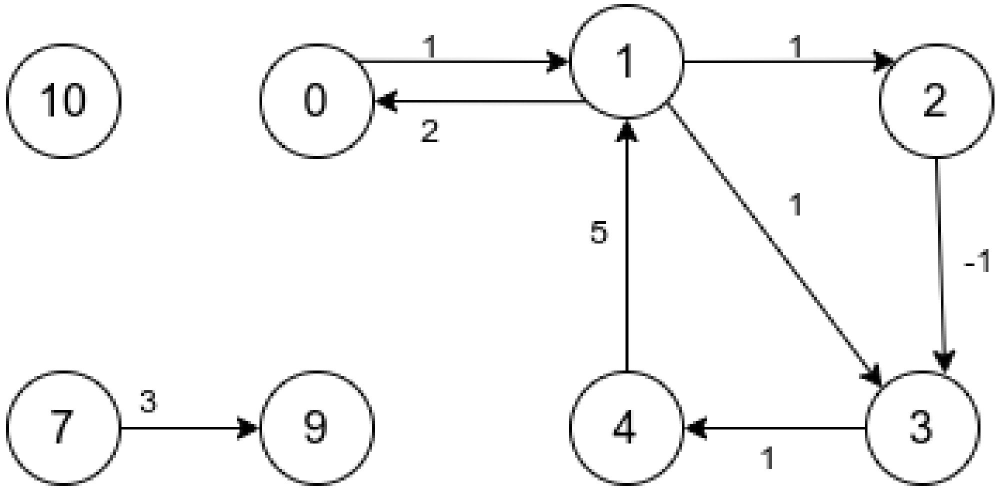
$$\sigma_{x.ID<50} (\bigcirc\ x)$$

Scaning and filtering is sequential. `Expand` is not.
$\Rightarrow$ `Expand` causes prefetch & cache misses.

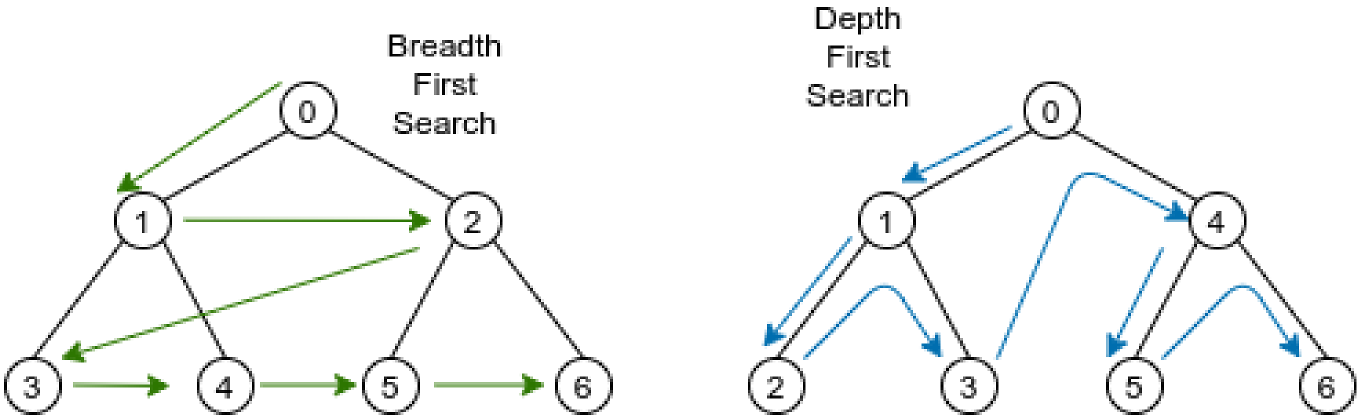
Example III

- Especially `Expand` jumps a lot. Potentially back and forth.
- Traversals rely primarily on expand.

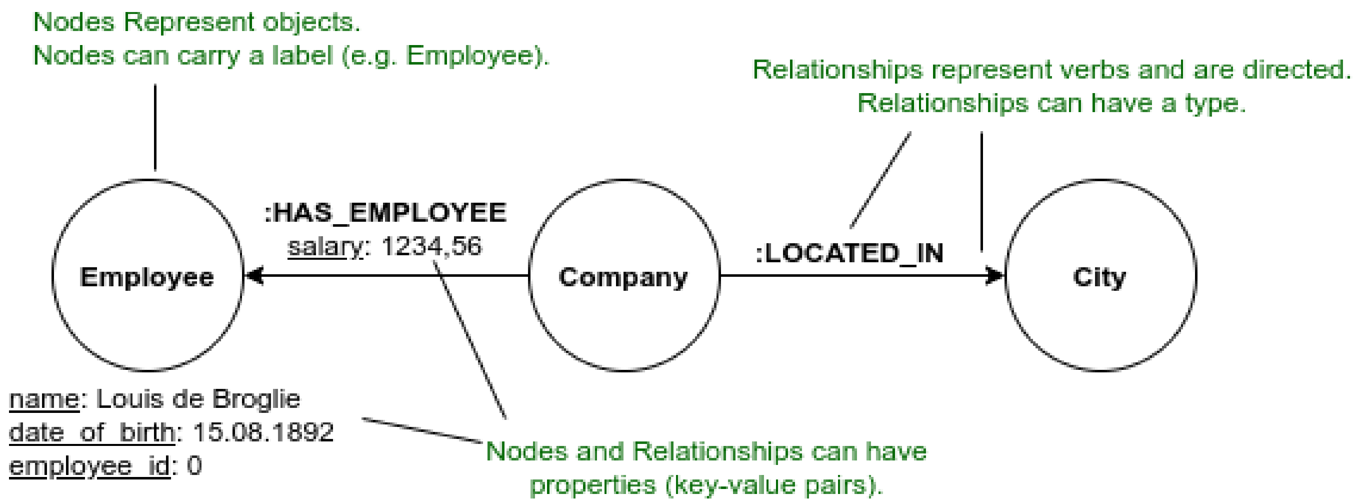
Graphs



Traversals



Property Graph Model



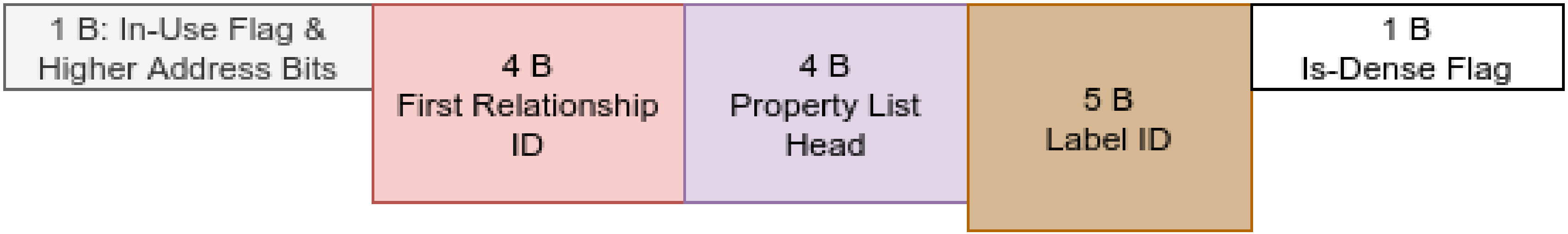
Data Structures I

Two essential record structures:

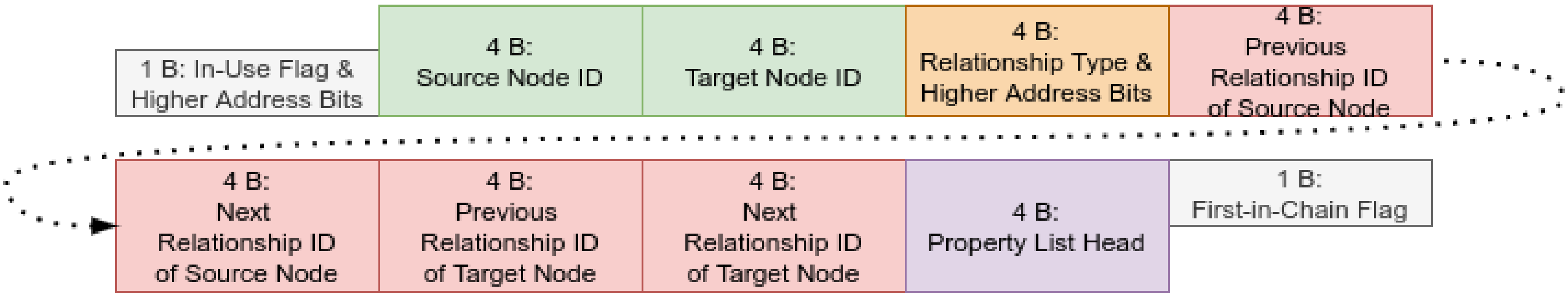
1. Node records

2. Relationship records

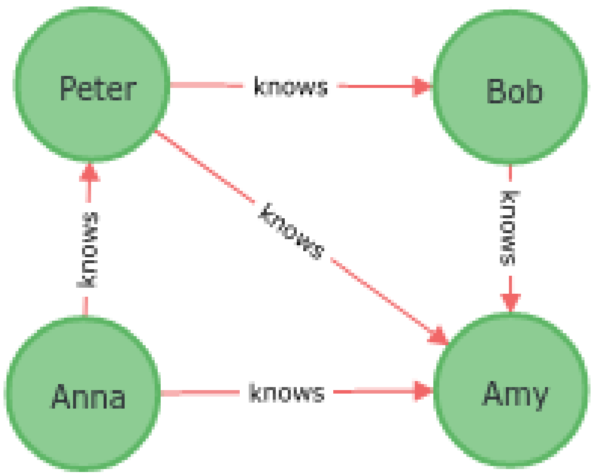
Data Structures II

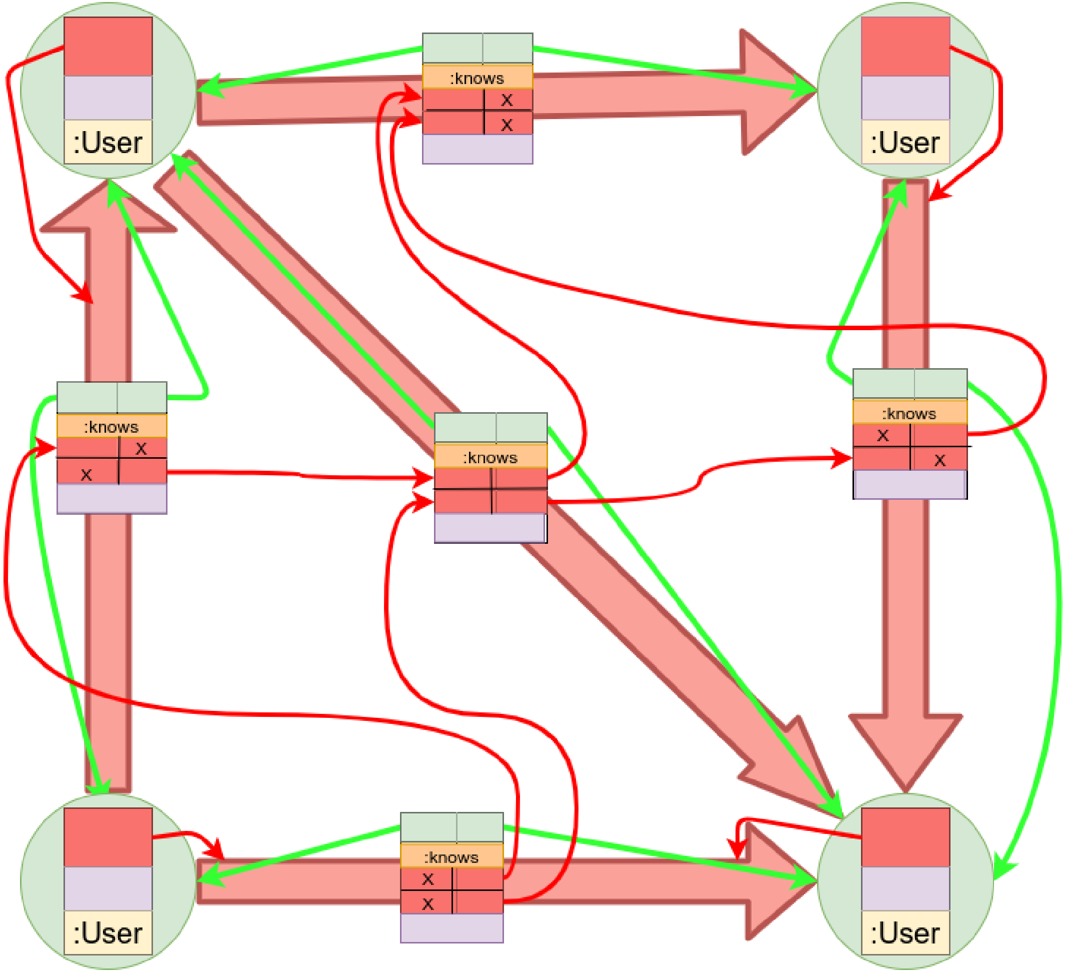


Data Structures III



Data Structures IV




:User
:knows
X
X
:User
:knows
X
X
:knows
:knows
X
X
:User
:knows
X
X
:User

Problem Definition I

Given a graph G, logical block size b, page size p.

Desired is

1. A partition of G into blocks of vertex records V_i and E_i relationship records,
2. permutations π_v, π_e of the blocks of vertex and edge records V_i, E_i,
3. a reordering of the incidence list pointers

such that spatial locality is as high as possible for traversal-based queries.

Problem Definition II

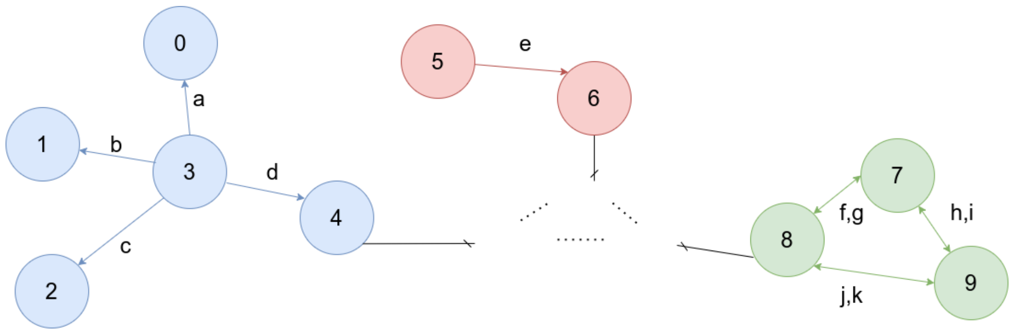
Temporal locality based on blocks.

$$P(X_{t+\Delta} = B | X_t = B)$$

Spatial locality in the same sense:

$$P(X_{t+\Delta} = B \pm \varepsilon | X_t = B)$$

Problem Definition III



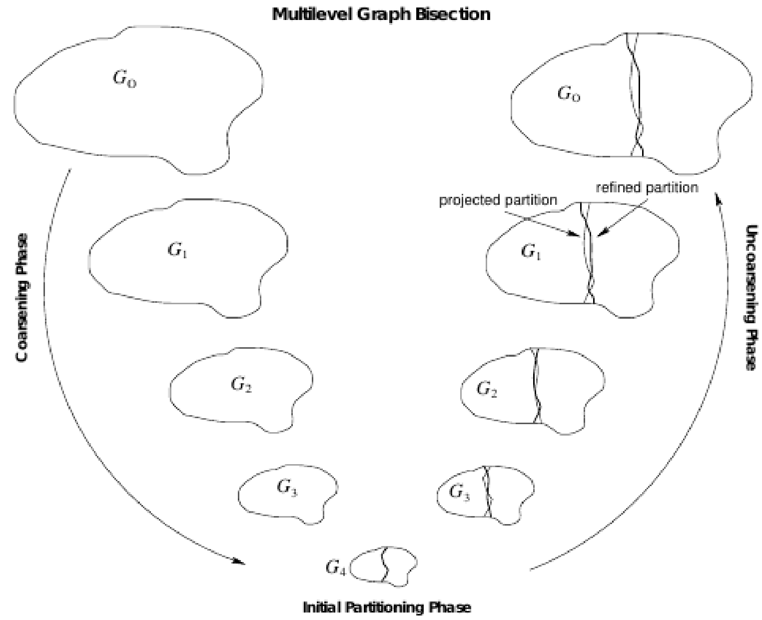
node.db	0, 5, 7	1, 4, 9	2, 6, 8	3		
edge.db	a, f	b, g	c, h	d, i	e, j	k

node.db	7,8, 9	0, 1, 3	2, 4, 5	6		
edge.db	f, h	g, k	i, j	a, b	c, d	e

Problem Definition IV

TODO insert incidence list chaos and ordered here

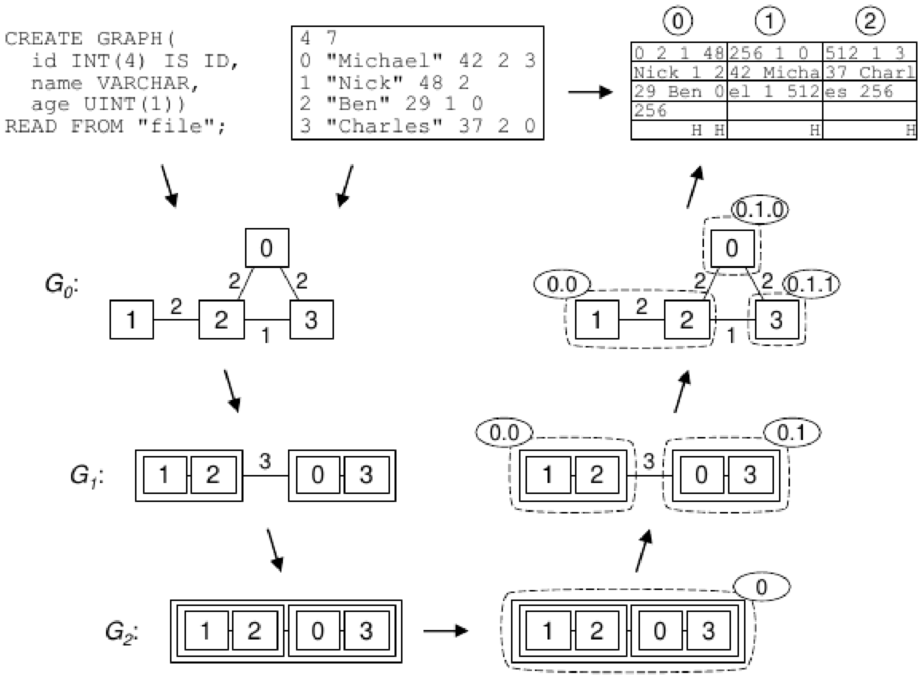
G-Store I


Multilevel Graph Bisection
G_0
G_1
G_2
G_3
G_4
G_0
G_1
G_2
G_3
projected partition
refined partition
Coarsening Phase
Uncoarsening Phase
Initial Partitioning Phase

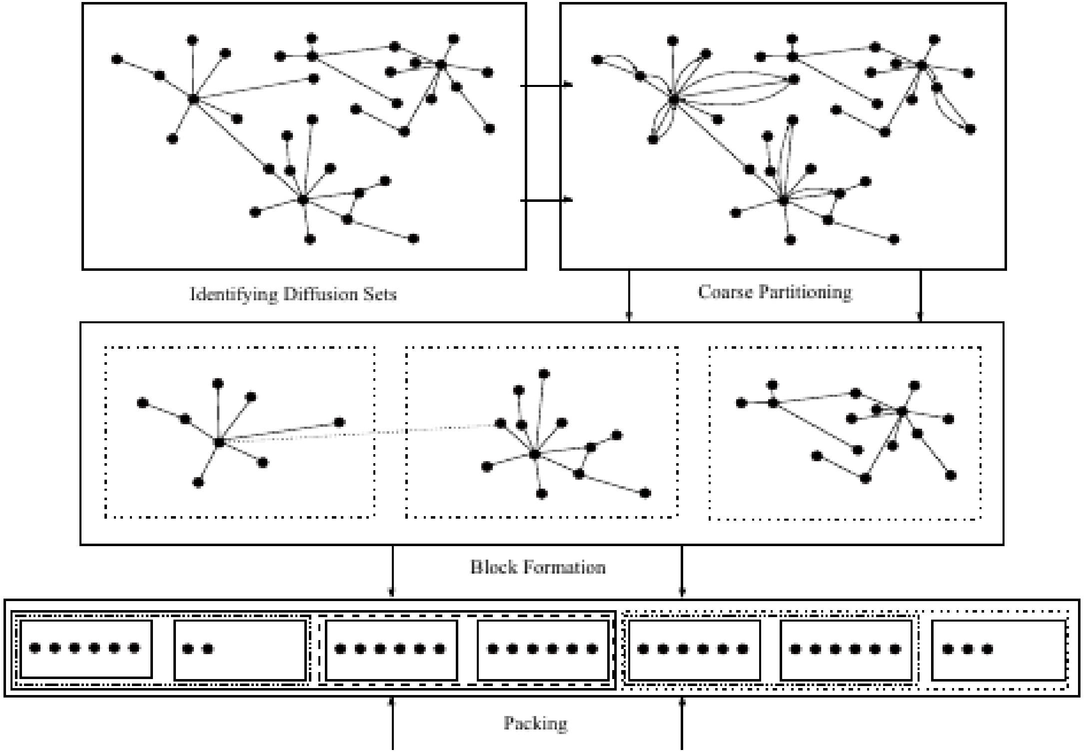
G-Store II

1. Coarsening: Heavy-Edge Matching
2. Turn-around
3. Uncoarsening
   - 3.1 Project
   - 3.2 Reorder
   - 3.3 Refine

G-Store III


CREATE GRAPH(
id INT(4) IS ID,
name VARCHAR,
age UINT(1))
READ FROM "file";
4 7
0 "Michael" 42 2 3
1 "Nick" 48 2
2 "Ben" 29 1 0
3 "Charles" 37 2 0
0 1 2
0 2 1 48
Nick 1 2
29 Ben 0
256
H H
256 1 0
42 Micha
el 1 512
H
512 1 3
37 Charl
es 256
H
G_0:
G_1:
G_2:
0.0
0.1
0.1.0
0.1.1

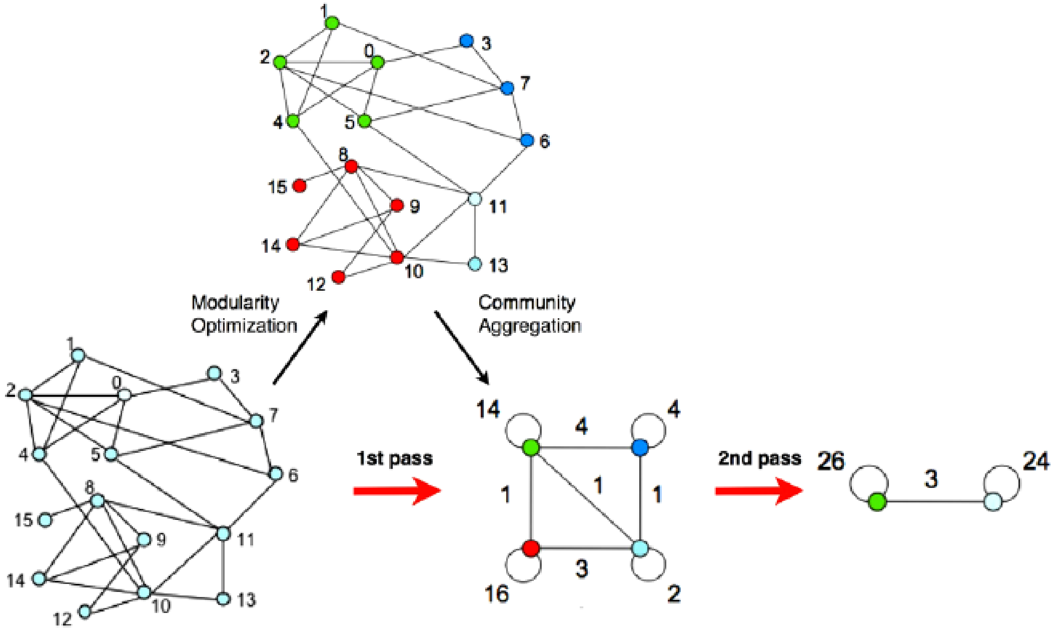
ICBL I


Identifying Diffusion Sets
Coarse Partitioning
Block Formation
Packing

ICBL II

- I Feature extraction: Do t random walks of length l.
- C Coarse clustering: Adapted K-Means.
- B Block Formation: Agglomerative hierarchical clustering.
- L Layout Blocks: Sort blocks and subgraphs

Louvain Method I



Louvain Method II

1. Initialize all nodes in singleton community.
2. Merge community into a neighboring community where modularity gain is maximal, until modularity gain is below threshold.
3. Construct new graph from aggregated communities and go to 1.

$$Q = \frac{1}{2m} \sum_{u,v \in V} \left(w_{(u,v)} - \frac{w_u w_v}{2m} \right) \cdot \delta(c_u, c_v)$$

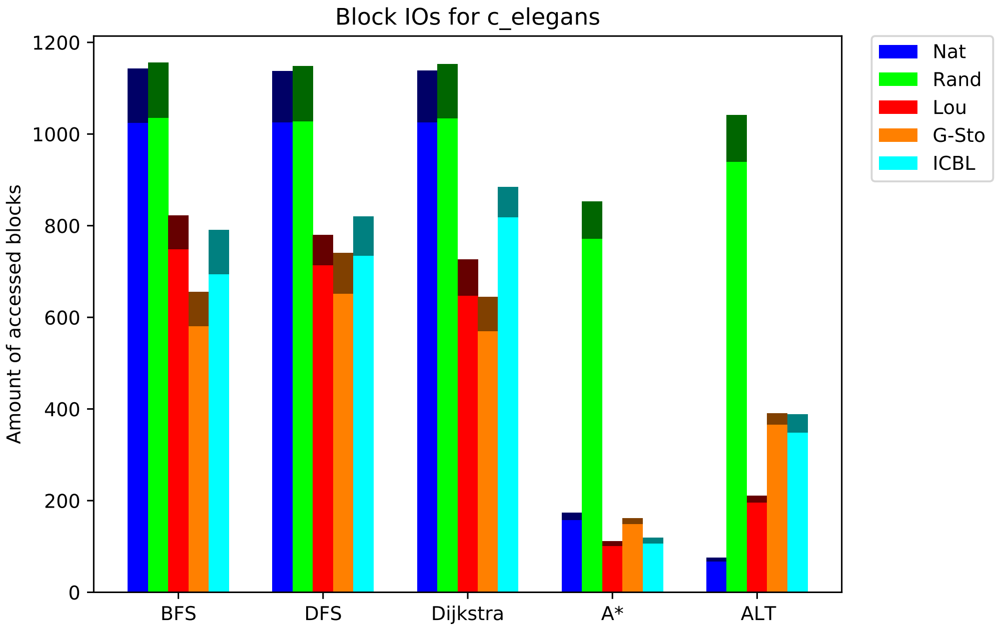
Incidence List Rearrangement

1. Traverse incidence list and store IDs.
2. Sort IDs.
3. Assign first relationship pointer to lowest ID.
4. Assign next pointer of new first relationship to second ID.
5. Assign next pointer of relationship i to $i + 1$. ID and prev pointer to $i - 1$. ID.
6. Assign next pointer of last relation to first and prev pointer of first to last.

Setup

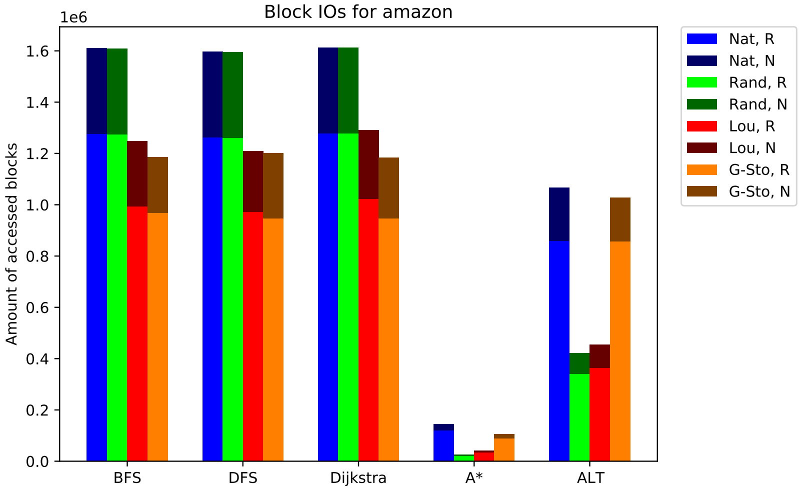
- Queries: BFS, DFS, Dijkstra, A*, ALT.
- Datasets: $[131, 1'134'890]$ nodes, $[764, 2'987'624]$ edges, average degree $[2.6, 25.5]$
- Domains include biological neural net, E-Mails, Co-authors, Frequent item sets, Comments.

Results I

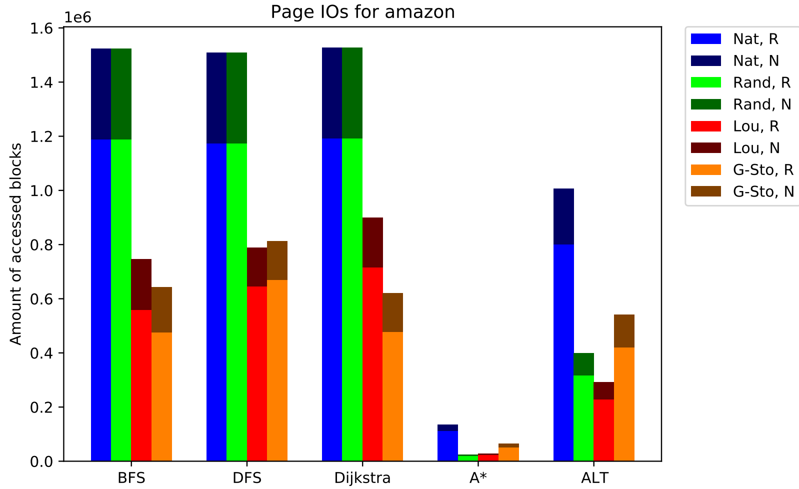


Block IOs for c_elegans
Amount of accessed blocks
0
200
400
600
800
1000
1200
BFS
DFS
Dijkstra
A*
ALT
Nat
Rand
Lou
G-Sto
ICBL

Results II

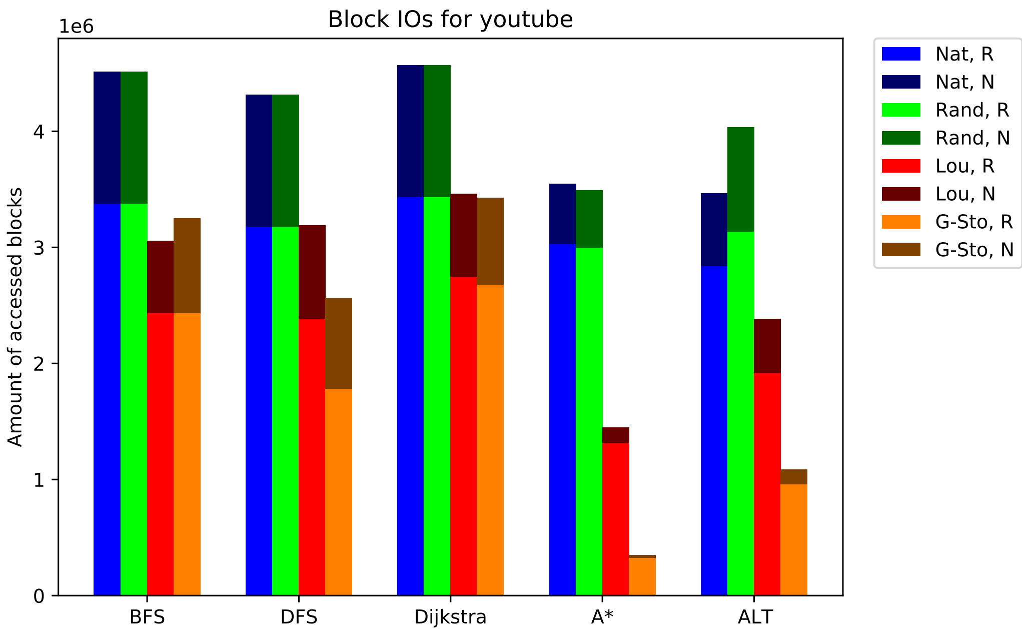


Block IOs for amazon
1e6
Amount of accessed blocks
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
BFS
DFS
Dijkstra
A*
ALT
Nat, R
Nat, N
Rand, R
Rand, N
Lou, R
Lou, N
G-Sto, R
G-Sto, N

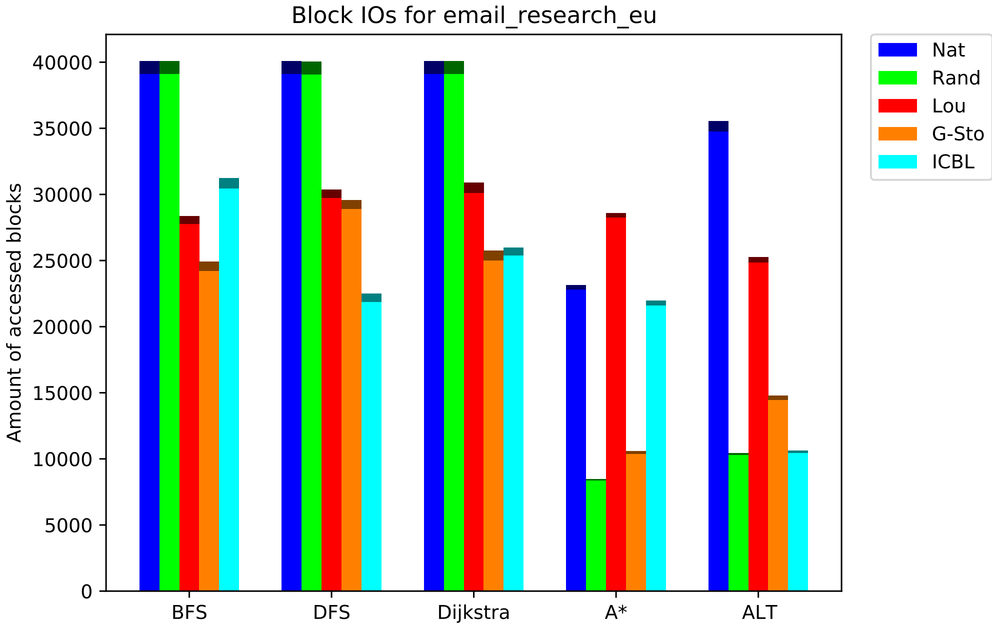


Page IOs for amazon
1e6
Amount of accessed blocks
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
BFS
DFS
Dijkstra
A*
ALT
Nat, R
Nat, N
Rand, R
Rand, N
Lou, R
Lou, N
G-Sto, R
G-Sto, N

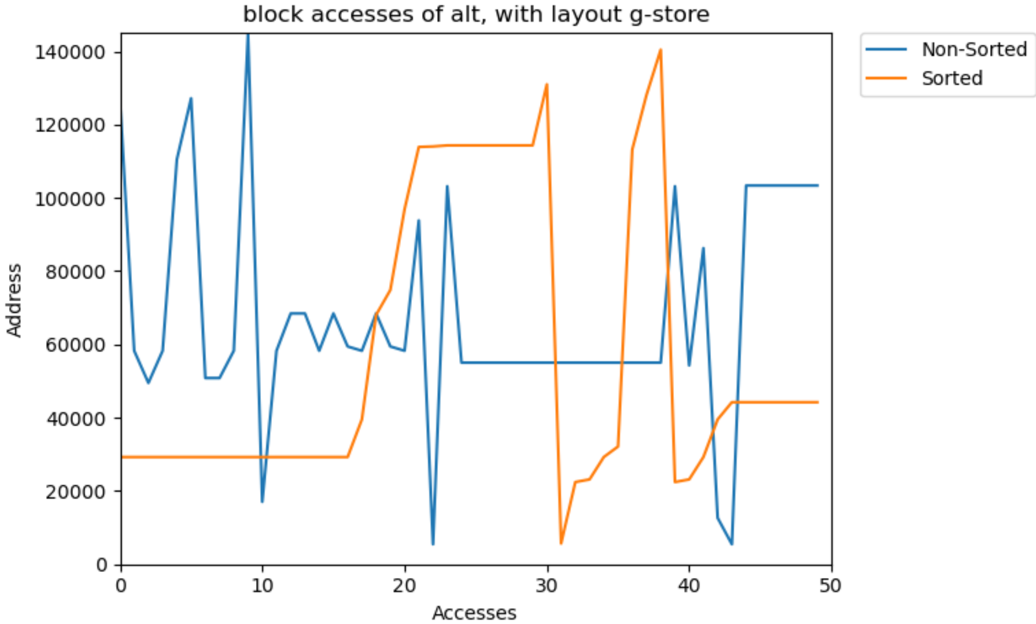
Results III


Block IOs for youtube
1e6
Amount of accessed blocks
0
1
2
3
4
BFS
DFS
Dijkstra
A*
ALT
Nat, R
Nat, N
Rand, R
Rand, N
Lou, R
Lou, N
G-Sto, R
G-Sto, N

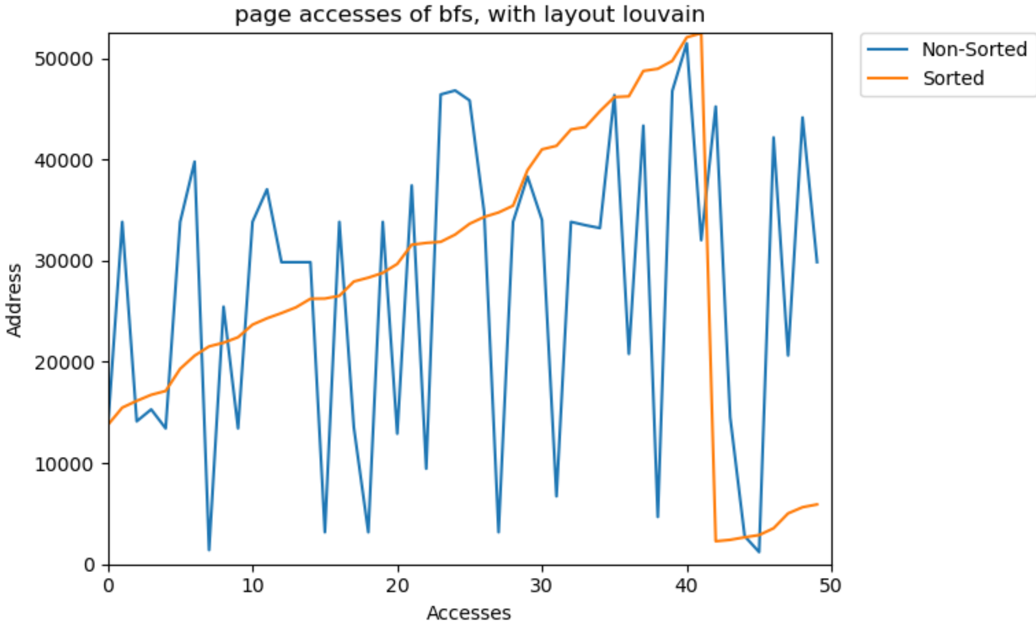
Results IV


Block IOs for email_research_eu
Amount of accessed blocks
40000
35000
30000
25000
20000
15000
10000
5000
0
BFS
DFS
Dijkstra
A*
ALT
Nat
Rand
Lou
G-Sto
ICBL

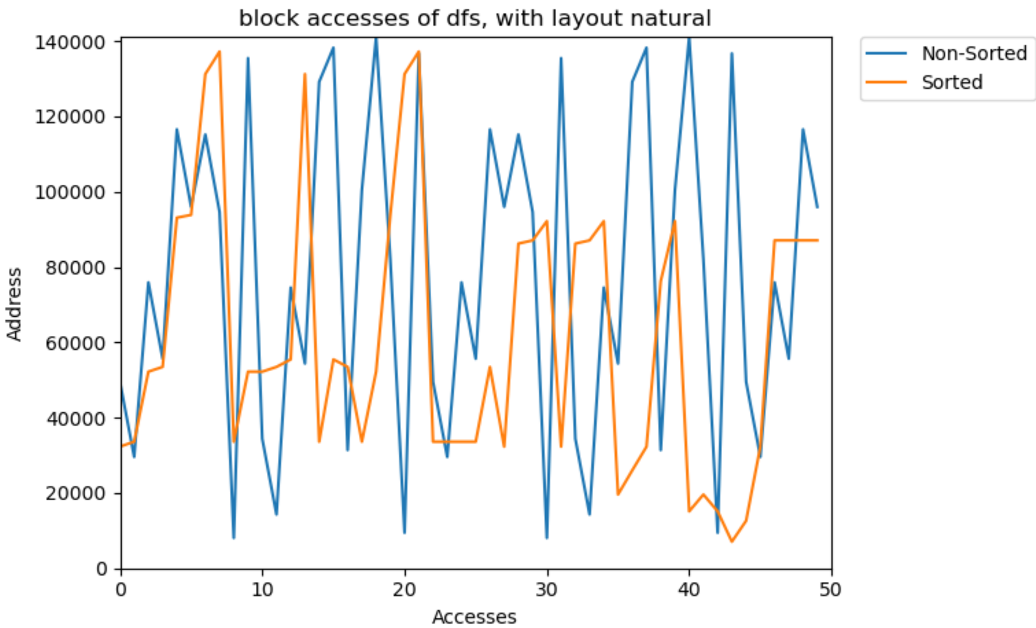
Results V


block accesses of alt, with layout g-store
Non-Sorted
Sorted
Address
Accesses
0
20000
40000
60000
80000
100000
120000
140000
0
10
20
30
40
50

Results VI


page accesses of bfs, with layout louvain
Address
Accesses
0
10000
20000
30000
40000
50000
0
10
20
30
40
50
Non-Sorted
Sorted

Results VII


block accesses of dfs, with layout natural
Non-Sorted
Sorted
Address
Accesses
0
20000
40000
60000
80000
100000
120000
140000
0
10
20
30
40
50

Summary I

- Static rearrangement methods decrease number of block accesses.
  $\Rightarrow$ increase locality
- Sorting the incidence lists leads to more sequential access sequences.
- Ordering the blocks is crucial for spatial locality.

Future Work I

- Leiden instead of Louvain
- RCM-based rearrangement
- Dynamic Rearrangement — Query-based
- Disk-based implementation