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

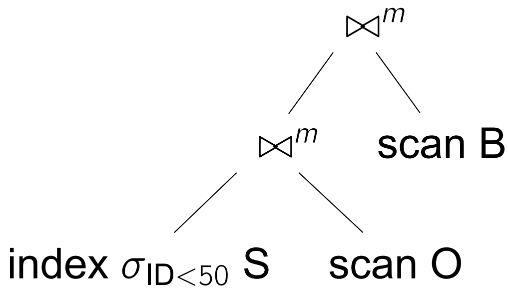
Motivation

Current state of performance-optimized

- relational databases: accesses are made as sequential as possible.
- graph databases: access is often 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”

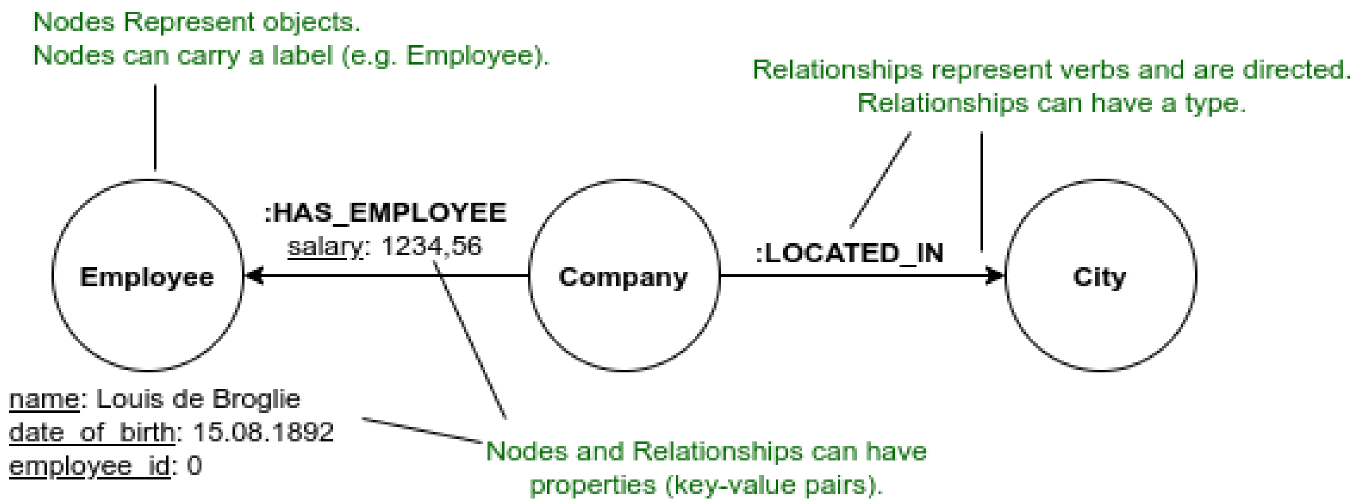
$$\begin{array}{c} \sigma_{y.\text{label = Boat}} \\ | \\ \uparrow_{x}^{y} \\ | \\ \sigma_{x.ID<50}\,(\bigcirc\ \mathrm{x}) \end{array}$$

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.

Property Graph Model



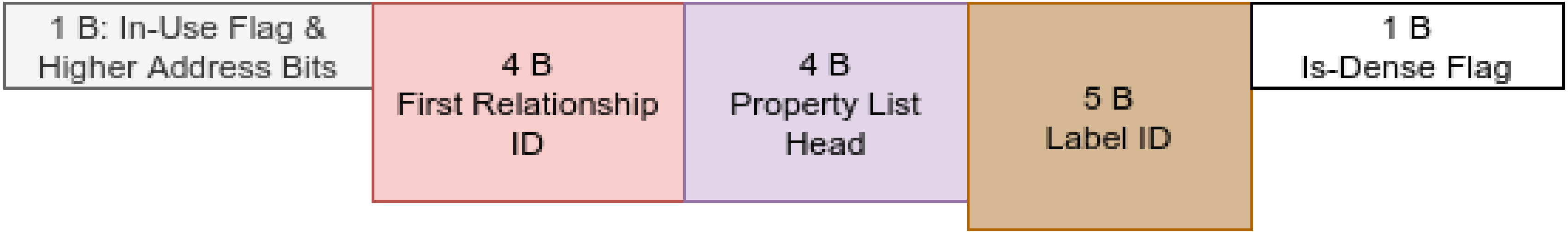
Data Structures I

Two essential record structures:

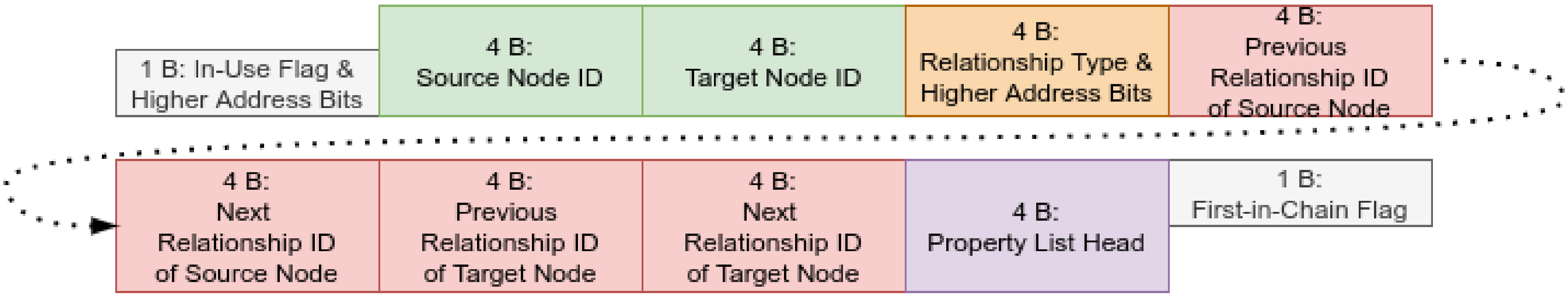
1. Node records
2. Relationship records

Inspired by Neo4J

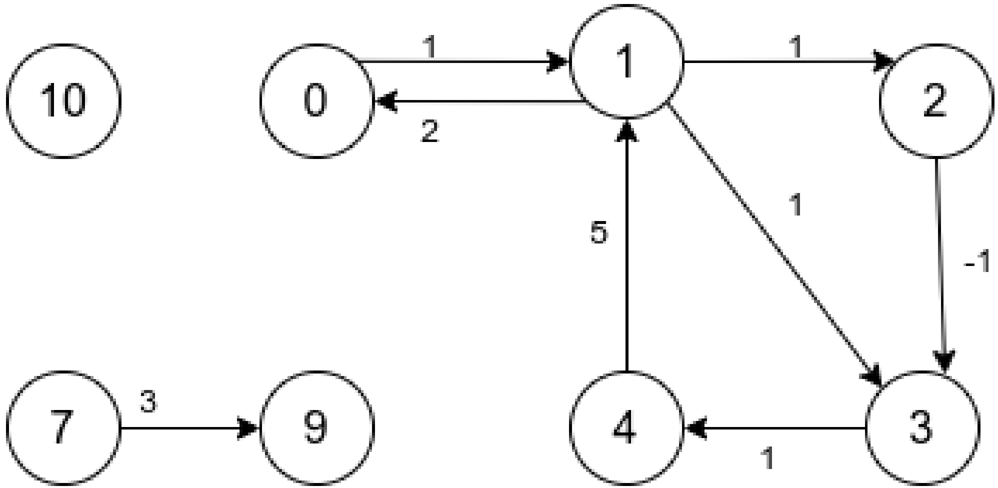
Data Structures II



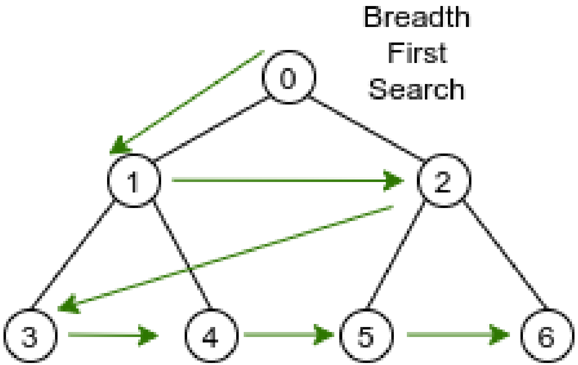
Data Structures III

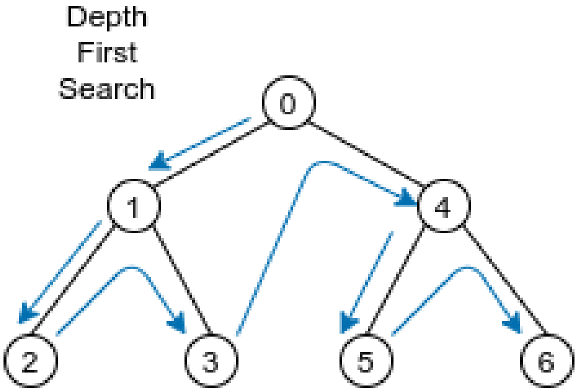


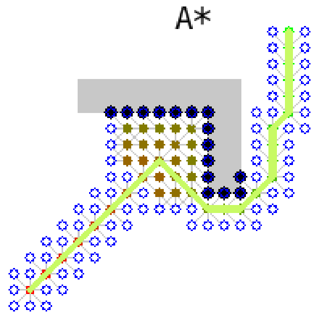
Graphs we focus on



Traversals







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

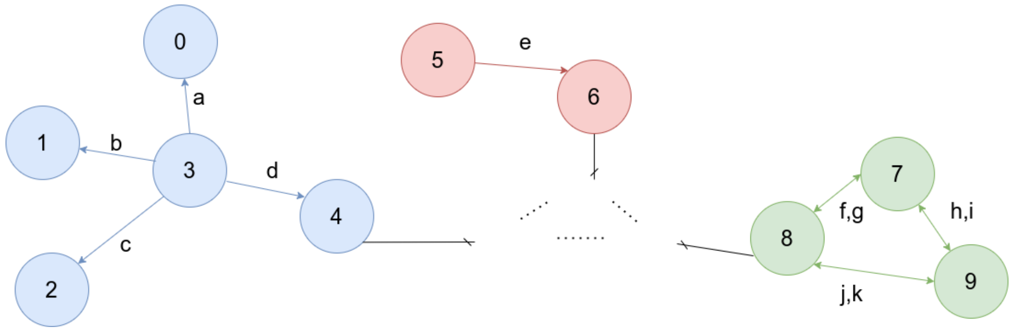
Temporal locality based on blocks.

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

Spatial locality in the same sense:

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

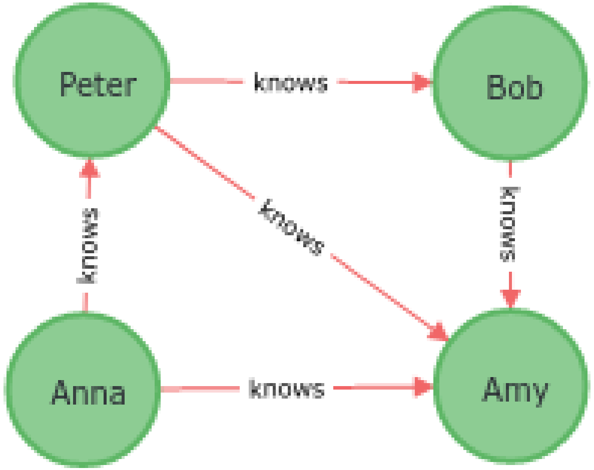
Problem Definition III



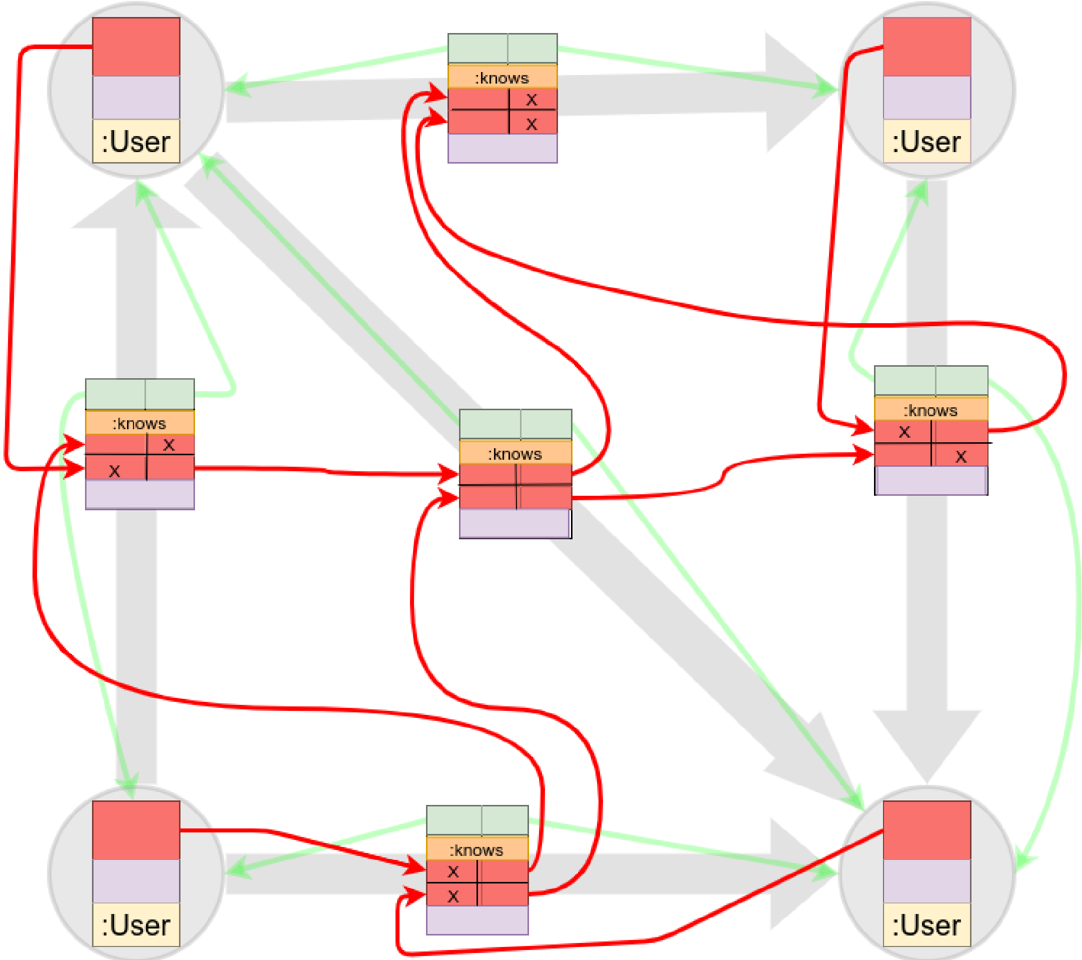
node.db	0, 5, 7	1, 4, 9	2, 6, 8	3		
edge.db	a, f	b, g	c, h	d, i	e, j	k

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

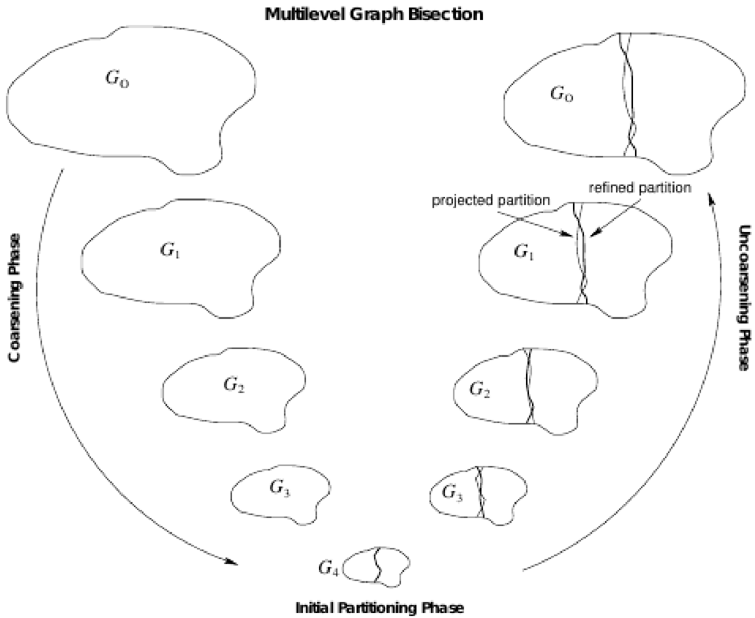
Problem Definition IV



3 Problem Definition



G-Store I

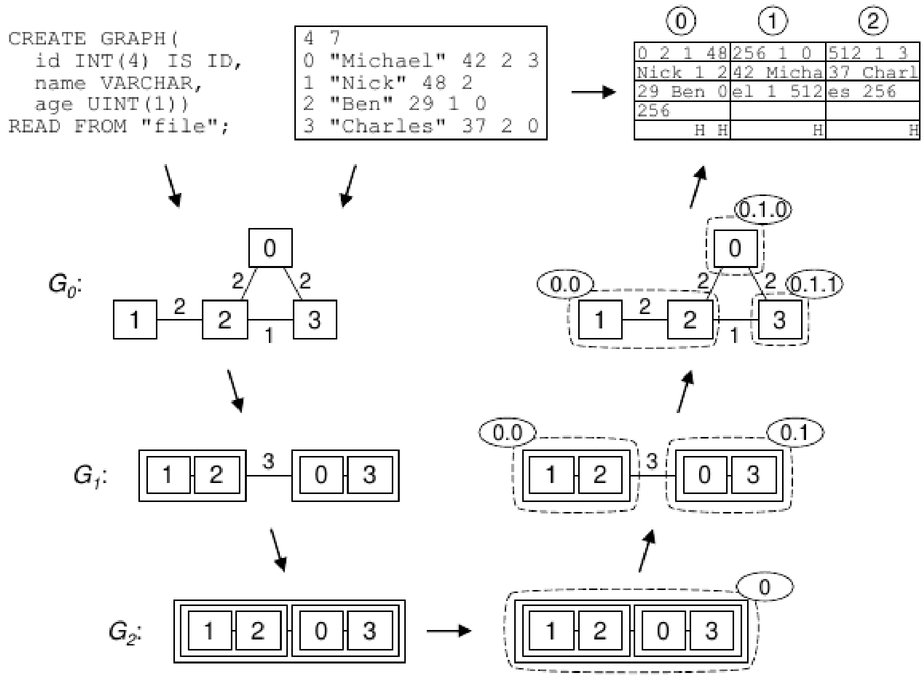

Multilevel Graph Bisection
G_0
G_1
G_2
G_3
G_4
G_0
projected partition
refined partition
G_1
G_2
G_3
Coarsening Phase
Uncoarsening Phase
Initial Partitioning Phase

G-Store II

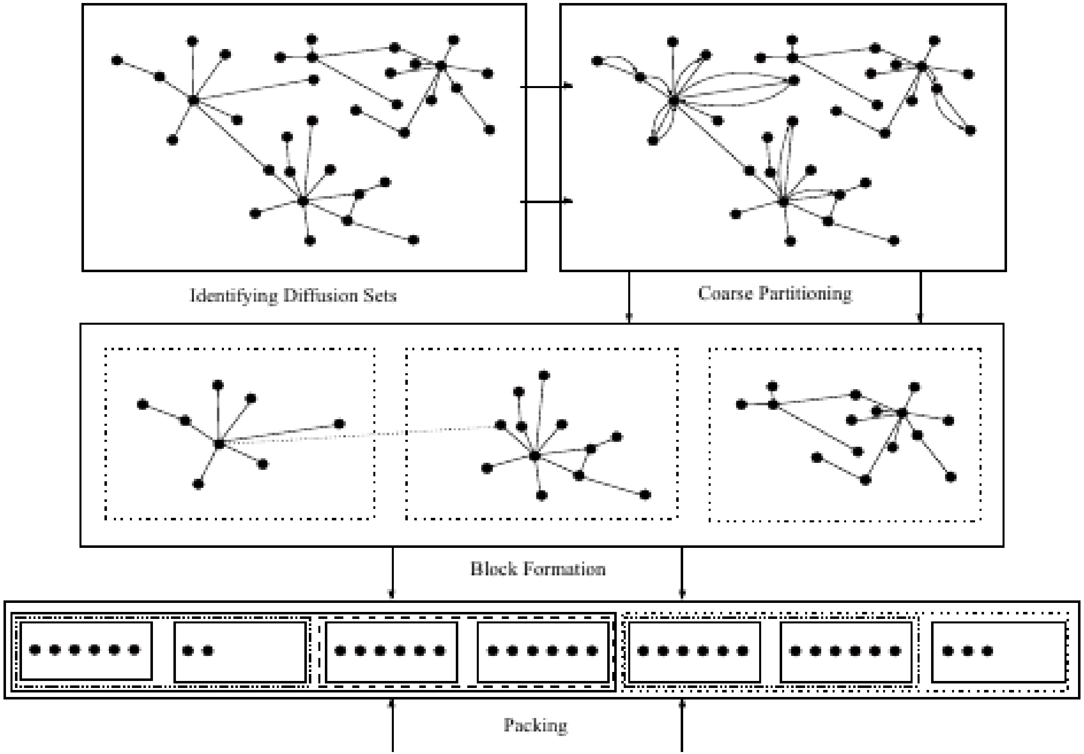
1. Coarsening: Heavy-Edge Matching
2. Turn-around
3. Uncoarsening
   3.1 Project
   3.2 Reorder
   3.3 Refine

$$\min \sum_{(u,v)\in E} |\phi(u) - \phi(v)|$$

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 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
0

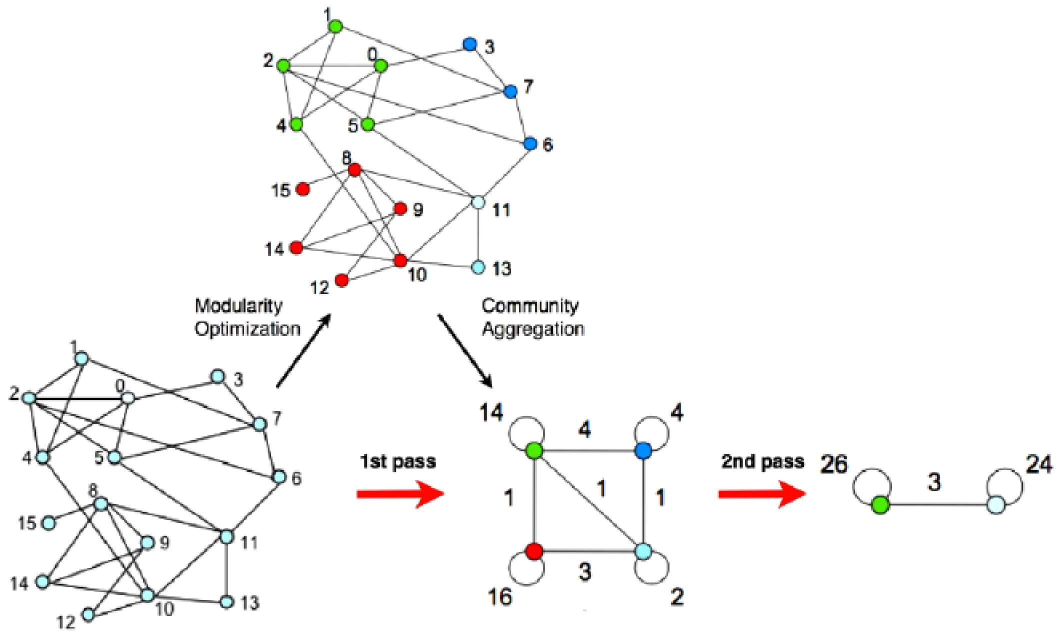
ICBL I


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

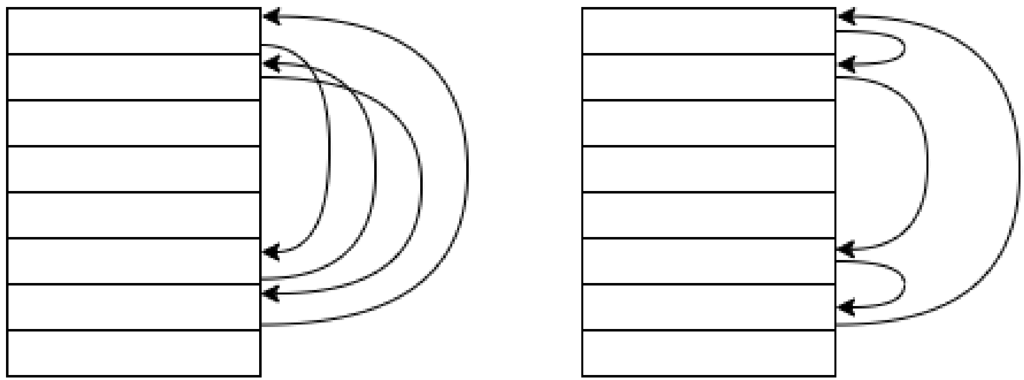


Louvain Method II

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.

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

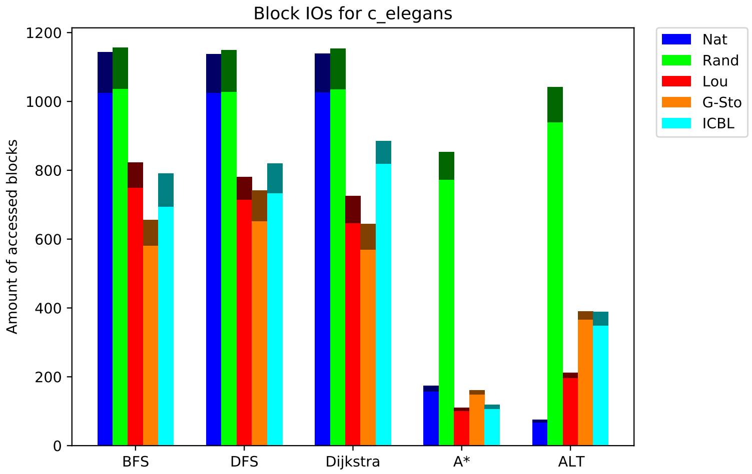
Incidence List Rearrangement

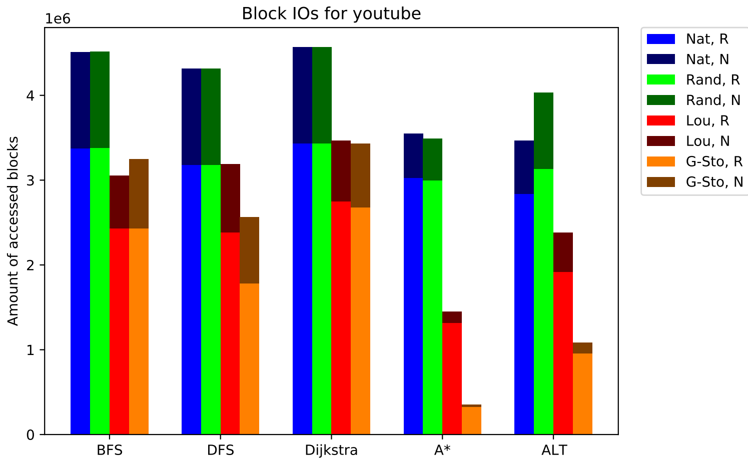


Setup

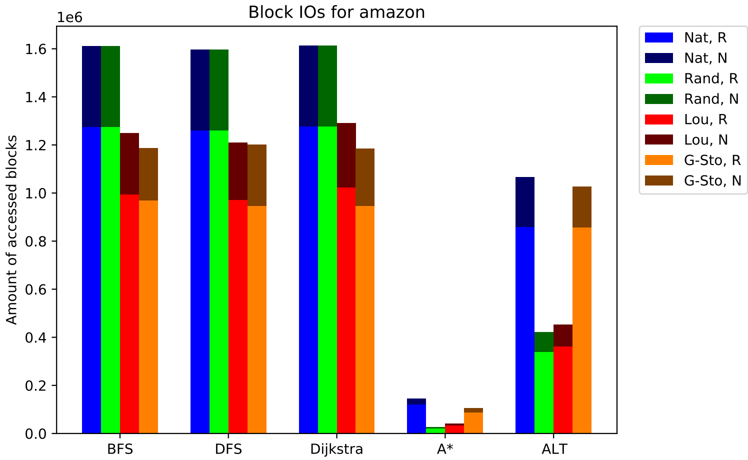
- Queries: BFS, DFS, Dijkstra, A*, ALT.
- Datasets: $[131, 1'134'890]$ nodes, $[764, 2'987'624]$ edges, average degree $[2.6, 25.5]$
- Domains include biological neural net, E-Mails, Co-authors, Frequent item sets, Comments.

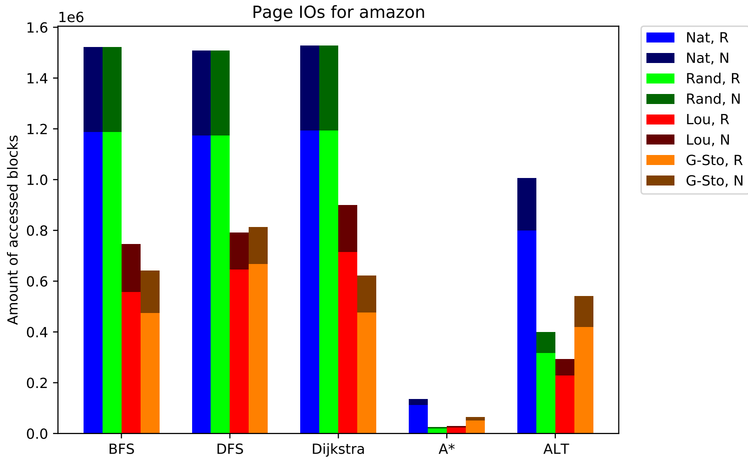
Results I


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

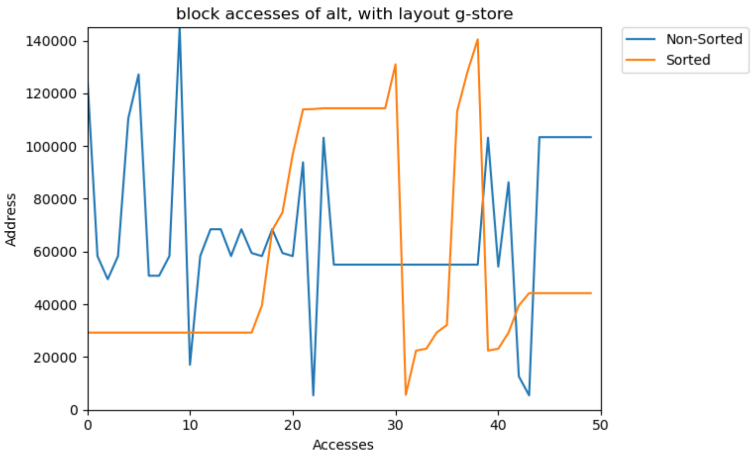

Block IOs for youtube
1e6
Amount of accessed blocks
0
1
2
3
4
BFS
DFS
Dijkstra
A*
ALT
Nat, R
Nat, N
Rand, R
Rand, N
Lou, R
Lou, N
G-Sto, R
G-Sto, N

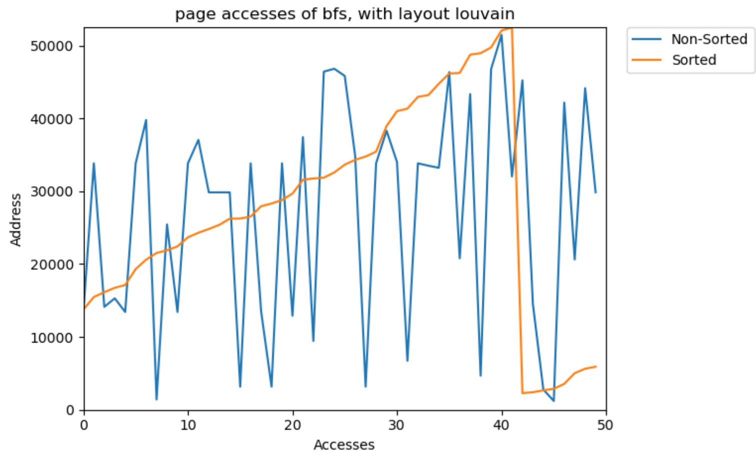
Results II


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

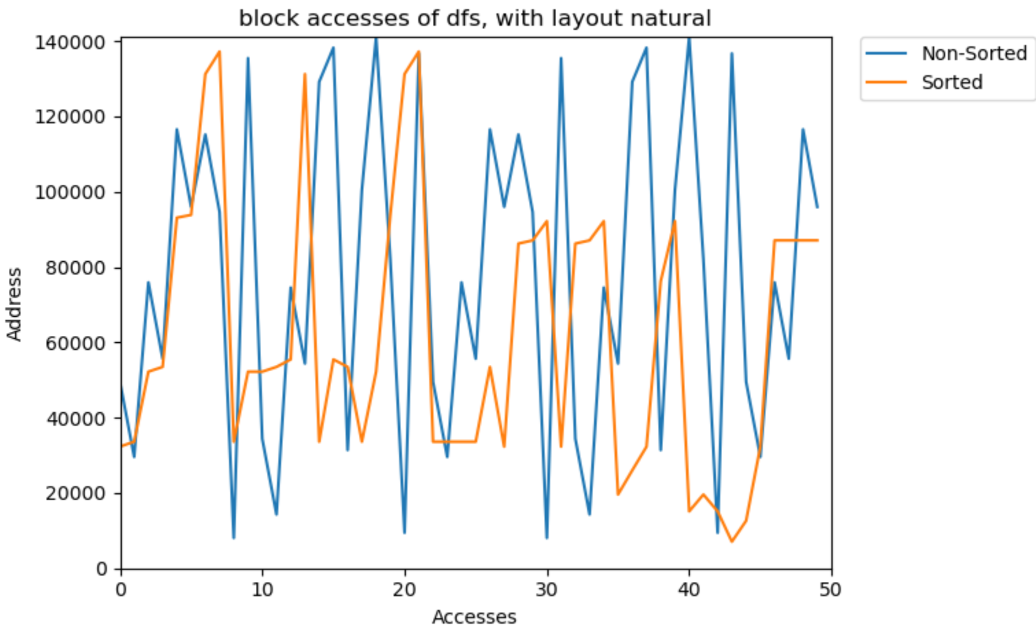

Page IOs for amazon
1e6
Amount of accessed blocks
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
BFS
DFS
Dijkstra
A*
ALT
Nat, R
Nat, N
Rand, R
Rand, N
Lou, R
Lou, N
G-Sto, R
G-Sto, N

Results III


block 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


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

Results IV


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

- 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

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