Optimizing Join Index Based Join Processing: A Graph Partitioning Approach

Shashi Shekhar, Chang-tien Lu, Sivakumar Ravada, Sanjay Chawla

Department of Computer Science

University of Minnesota

Presented by: Chang-tien Lu

Overview

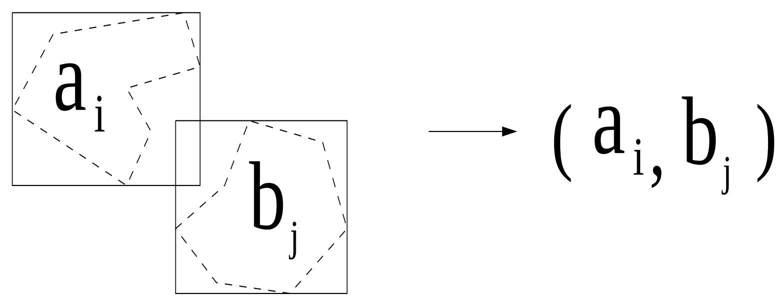
- Motivation
- Basic Concept
- Asymmetric Spatial Clustering
- Symmetric Spatial Clustering
- Experiment Design and Result
- Conclusion and Future Work

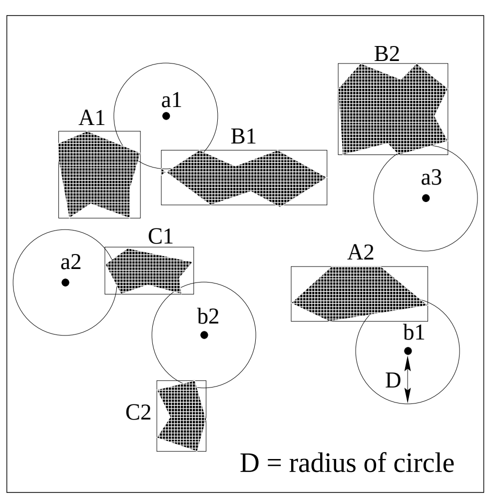
Motivation

- Spatial data is complex
  - *often contains points, lines, polygons, etc.*
- Spatial data is large
  - *often contains polygons with thousands of edges*
- Spatial data processing is expensive
  - *a typical query on a 100,000 polygon data set takes 200 seconds*
- Research questions
  - *How to efficiently process a range query?*
  - *How to efficiently process a join query?*

Spatial Join

- A join between two sets of spatial objects
- $MBR(a_i)$: The minimum bounding rectangle of the spatial object a_i.
- Input: Two sets of spatial objects $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_m)$
- Output:
  - *All pairs* $(Id(a_i), Id(b_j))$ *with* $MBR(a_i) \cap MBR(b_j) \neq 0$





Join-index based Join processing

- What is join index
  - *Describe a relationship between two relations*
  - *Each tuple identifies one tuple in the result*
  - *Pre-joined relation*
- Benefits
  - *Much smaller than joined relation*
  - *Simple structure*
  - *Compatible with other operations, e.g., select and union*
- Applications
  - *Low update rate, quick response, large data set*
  - *Example: Spatial data set*

R Relation
(Facility Location)

ID	Location (X,Y)	Non-Spatial Data
a1	(7.9,16.7)	(...)
a2	(3.4,11.4)	(...)
a3	(19.5,13.1)	(...)
b1	(18.7,6.4)	(...)
b2	(9.5,7.1)	(...)

S Relation
(Forest-Stand Boundary)

ID	MOBR $(X_{LL}Y_{LL}X_{UR}Y_{UR})$	Non-Spatial Data
A1	(3,12.2,6.6,16)	(...)
A2	(13.4,7.7,19.5,10)	(...)
B1	(7.6,12.7,15,5.2)	(...)
B2	(15.5,15,20.4,19)	(...)
C1	(5.1,8.9,9.1,10.9)	(...)
C2	(7.5,2,9.7,15.1)	(...)

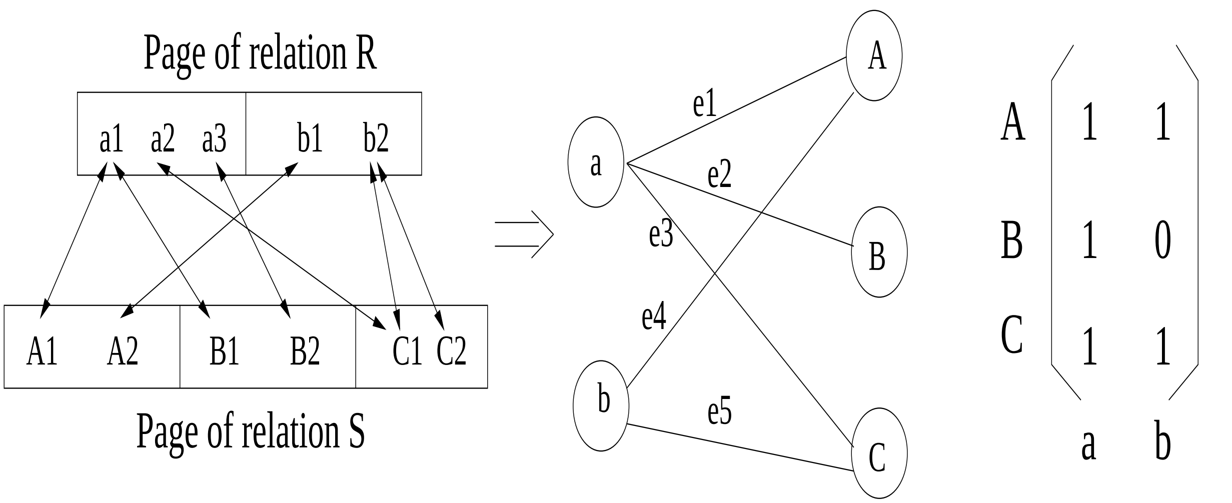
Join Index

R.ID	S.ID
a1	A1
a1	B1
a2	C1
a3	B2
b1	A2
b2	C1
b2	C2

Join index and Page Connectivity Graph (PCG)

- What is PCG
  - *A PCG is a bi-partite graph*
  - *A PCG describe the join-index in the form of a graph*

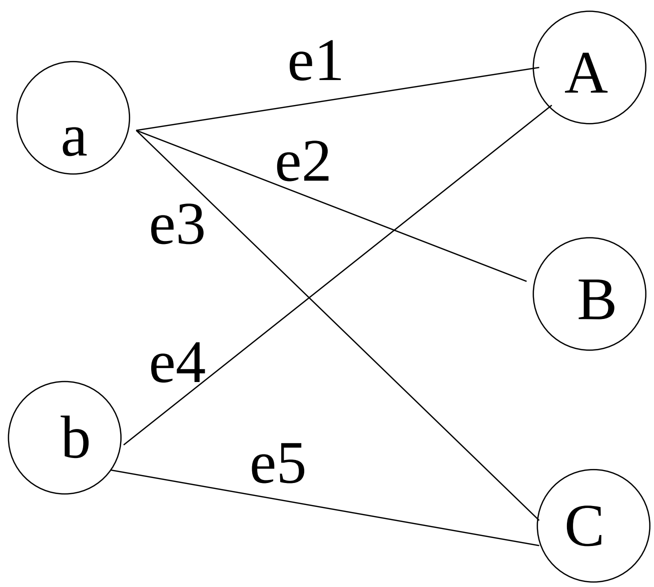
- Construction of PCG



Join Index PCG Adjacency Matrix

Page Access Sequence

- Example
  - *Assume buffer space can hold two pages of base relations*
  - *Consider the two page access sequence*
    - (a, A, b, B, a, C, b): total 7 page accesses
    - (a, A, b, C, a, B): total 6 page accesses
  - *Optimal number of page accesses = 5*



Page Access Sequence Problems

- Optimal page access sequence with fixed buffer (OPAS-FB)
  - **Given:** *A page-connectivity graph and a buffer of size B.*
  - **Find:** *A page-access sequence.*
  - **Objective:** *Minimize the number of page accesses*
  - **Constraint:** *Number of pages in buffer* $\leq$ *B.*
- Example
  - *Buffer size = 2*
  - *Page access sequence: a, A, b, C, a, B*
- OPAS-FB problem is NP-hard.
- Techniques for solving the problem
  - *Cluster all the related pages*
  - *Reuse pages in memory*

Asymmetric Spatial Clustering: Motivation

- Sorting
  - No redundant on primary relation; Redundant access to the second relation
- AGP
  - Reduce redundant access to the second relation
  - Example: 49 nodes PCG (R relation: 1-24, S relation: A-Y)

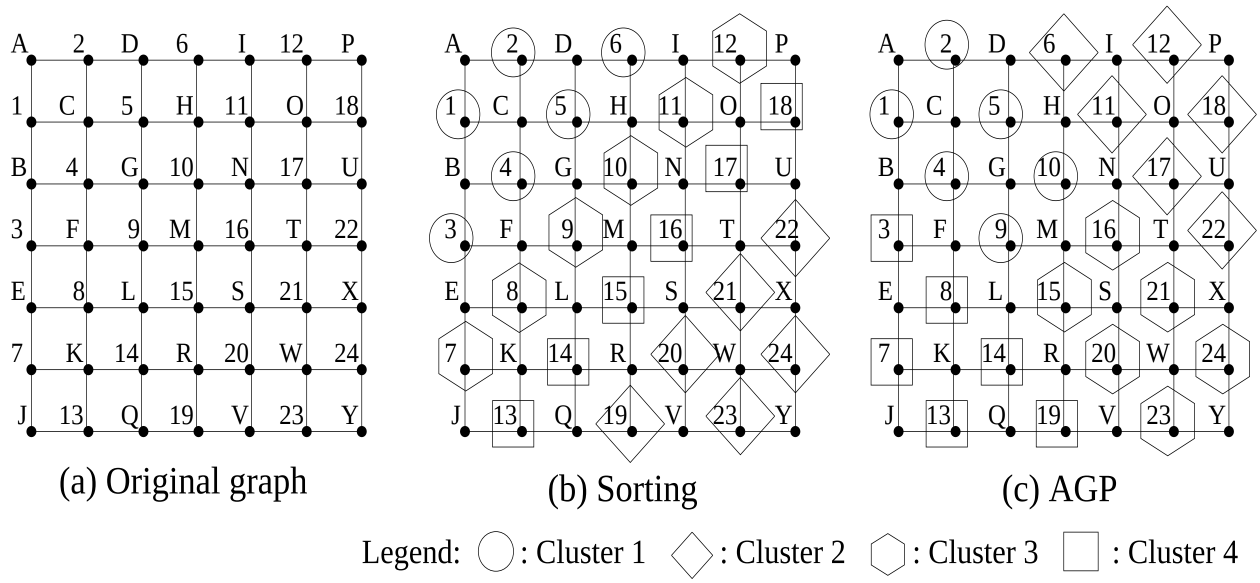


Figure 1: Example of AGP and Sorting heuristic

	Sorting Heuristic		**AGP**	
Round	First Six Buffers	The 7th buffer	First Six Buffers	The 7th buffer
1	1,2,3,4,5,6	A,B,C,D,E,F,G,H,I	1,2,4,5,9,10	A,B,C,D,F,G,H,L,M,N
2	7,8,9,10,11,12	E,F,G,H,I,J,K,L,M,N,O,P	3,7,8,13,14,19	B,E,F,J,K,L,Q,R,V
3	13,14,15,16,17,18	J,K,L,M,N,O,P,Q,R,S,T,U	6,11,12,17,18,22	D,H,I,N,O,P,T,U,X
4	19,20,21,22,23,24	Q,R,S,T,U,V,W,X,Y	15,16,20,21,23,24	L,M,N,R,S,T,V,W,X,Y
Total I/O	**66**		**62**	

Table 1: Loading Sequence of Sorting and APG

AGP Algorithm

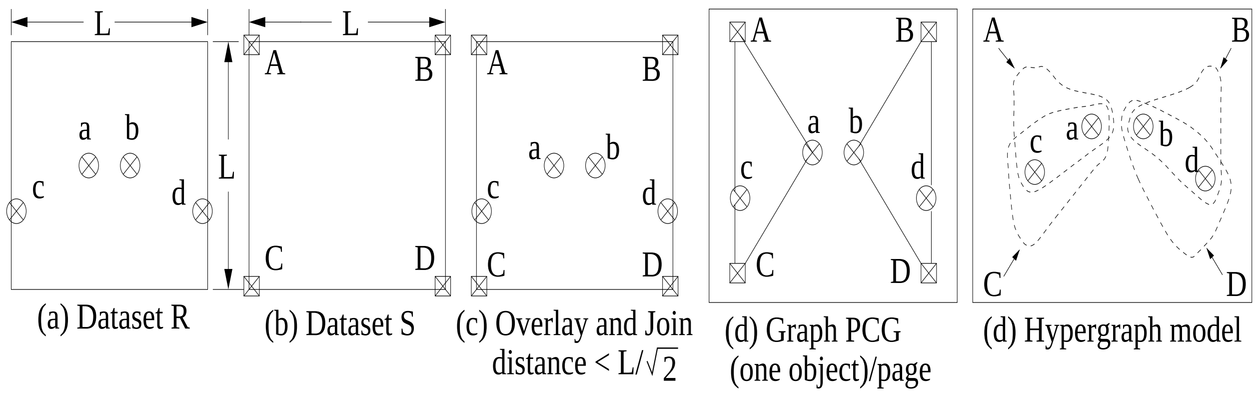


Figure 2: Construction of a one-side hyper graph from the data set

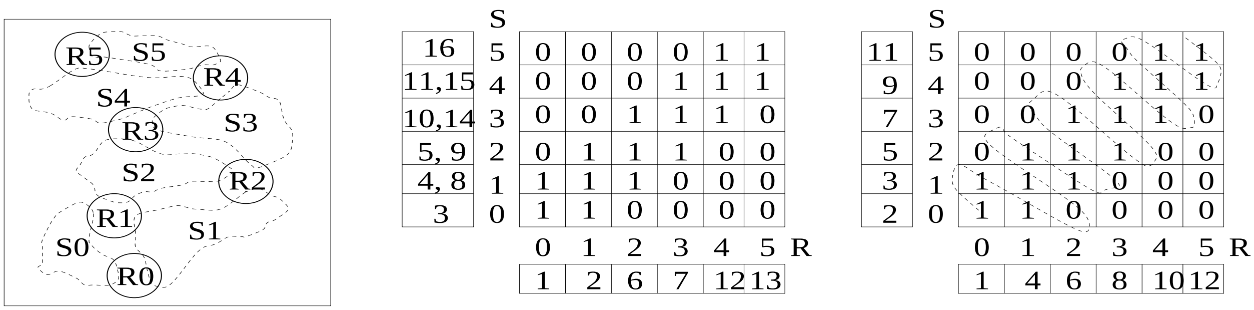
AGP Algorithm

Input: $G = (V_r, V_s, E)$ is a page connectivity graph

Output: $S =< P_1, P_2, ..., P_r >$ is a page access sequence with $r \geq |V_r| + |V_s|$.(P_is need not be distinct)

assert($|V_r| < |V_s|$);
assert($B \geq 2$); /* *number of buffers* */
$HG_r(V_r, HE_r)$ = DeriveHypergraph(G); /* HG_r *is a hypergraph*, $|HE_r| = |V_s|$ */
/* *For each node in* $|V_s|$, *build a hyperedge to encompass all of its corresponding nodes in* V_r */
$PSet_r$ = hMetis-Partition($HG_r, B-1$) /* *Minimize the number of hyperedge-cut set* */
i=0;
while ((P_{i_r}=SelectUnprossedPartition($PSet_r$))!=NULL) /* *Select the un-processed partition* */
{
  AddPageSequence(S, P_{i_r});/* *Add all the pages in* P_{i_r} *into the loading sequence* */
  P_{i_s} = Sort-Eliminate-Dup(G, P_{i_r});
  /* *Sort and eliminate the duplicated pages in* V_s *of* G *which connect to* P_{i_r} */
  AddPageSequence(S, P_{i_s});/* *Add all the pages in* P_{i_s} *into the loading sequence* */
  P_{i_r}.flag = "processed"; /* *Mark this partition as "processed"* */
  i++;
}

Motivation for Symmetric Method



(a) Spatial Join (b)Sorting/AGP (c)Symmetric method

Figure 3: Comparison of symmetric and asymmetric methods

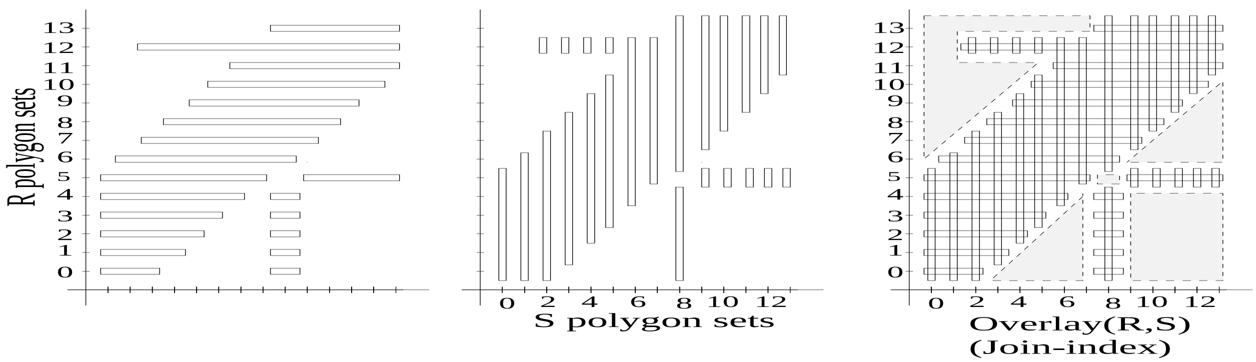


Figure 4: Overlay of two relations

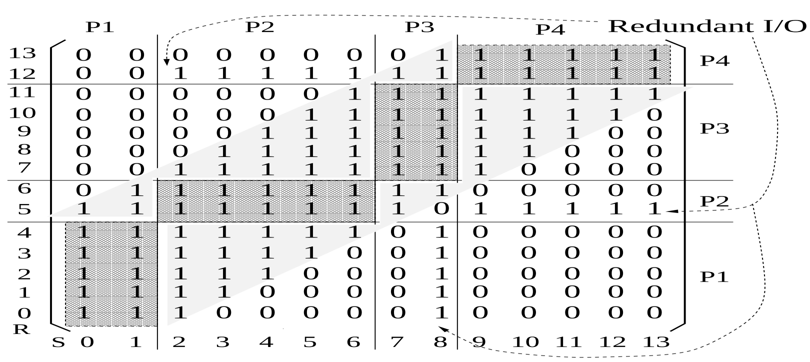


Figure 5: PCG for Join of Two Relations, with partition used by SGP

Comparison for all Symmetric methods

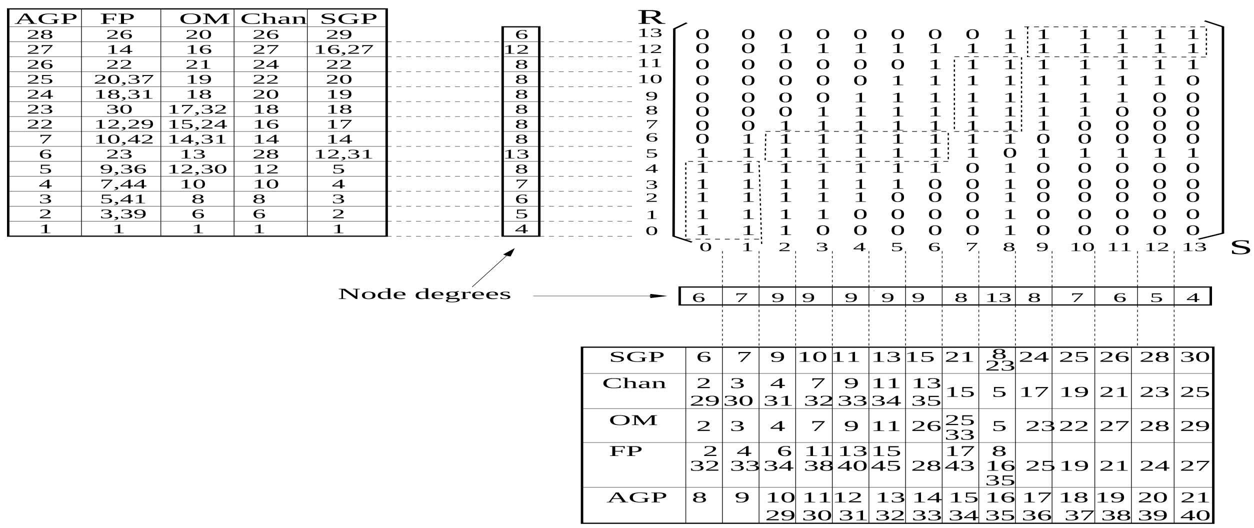


Figure 6: Comparison of different methods

Method		AGP/Sorting	FP	OM	Chan	SGP
Total I/O		40	45	33	35	31
Nodes with redundant I/O		S2-S13	R1-R4, R6, R7, R9, R10, S0-S5, S7, S8	R4, R6-R8, S7	S0-S6	R12, R5, S13
Degree of nodes with redundant I/O	Avg	8	8.3	8	8.3	12.6
	Min	4	5	8	6	12
	Max	13	13	8	9	13

Table 2: Summary of page access for different algorithms

- Symmetric Method
  - FP, OM, Chan: incremental, consider only local information in PCG
  - SGP: exploit global information across entire PCG

Proposed Approach 2: Symmetric Spatial Clustering

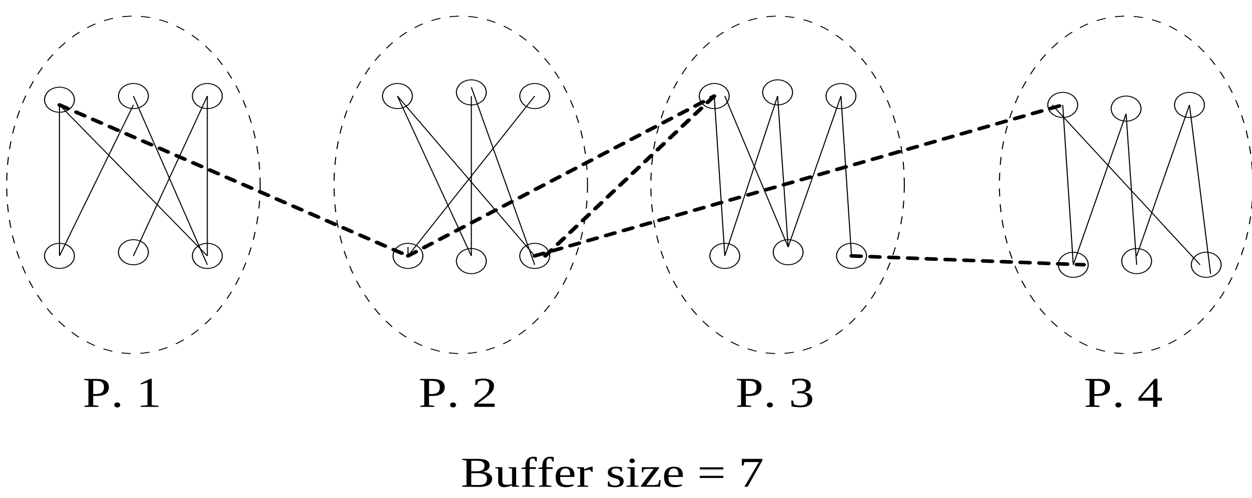
- Framework
  - *Given: A page connectivity graph in adjacency matrix* M_{PCG} *form and buffer size B to hold pages of R and S.*
  - *Find: A permutation of rows and a permutation of columns of* M_{PCG}*.*
  - *Objective: Minimize*

$$\sum_{V_i \in (vertx\ cover\ of\ outside\ B\ diagonal\ edges)} \lceil \frac{number\ of\ outside\ B\ diagonal\ edges\ incident\ on\ V_i}{B} \rceil \quad (1)$$

  - *Where: outside B-diagonal edge* $(M[i,j]) = 1$ *iff* $|i - j| \leq \lfloor \frac{B}{2} \rfloor$
  - *Constranit: Memory buffer* $\leq B$

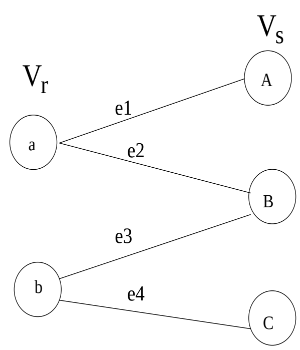
Symmetric Graph Partitioning(SGP) Heuristic

- P=partition(G,B-1): minimizes the number of edge-cut set.
- Choose a partition at random. Call this partition P_m
- Bring all of the pages in P_m into memory
  - *Perform the join within the P_m*
  - *Perform the cut-edge join outside the P_m*
  - *Mark the cut-edge "PROCESSED"*
- Choose another unprocessed partition, do the same

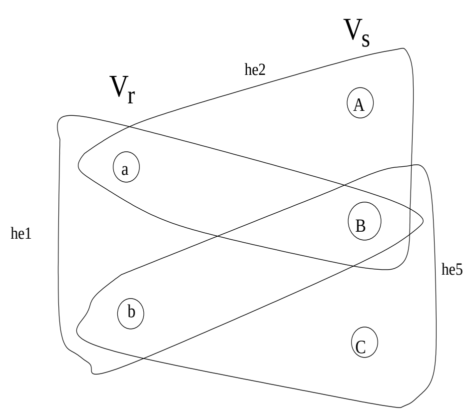


Refinement 1 for SGP: Reducing Cut Edges

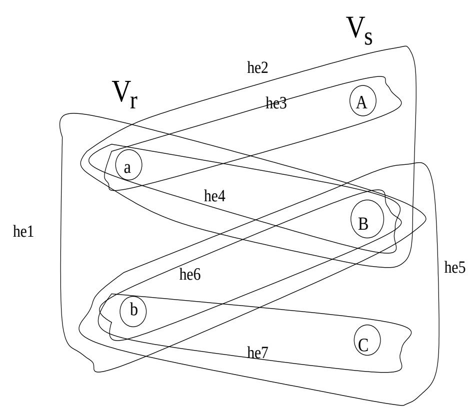
- Build the hyper graph based on the bipartite graph
- Use hypergraph partition algorithm, hMetis, to partition this hypergraph
- Savings = $\max(E_c - E_h, 0)$.



Original bipartite graph



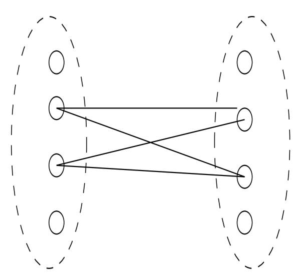
Pure hyperedge graph



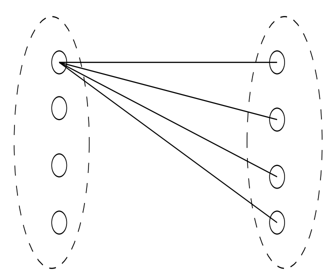
Hyperedge graph

Refinement 2 for SGP: Partitions Scheduling and Vertex Cover Strategy

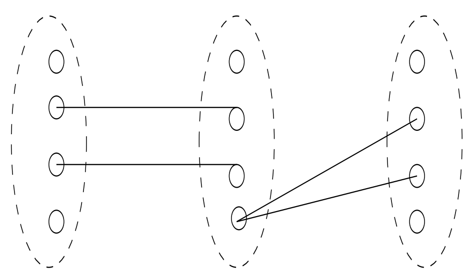
- Goal: Reduce the redundant I/O by properly scheduling their processing
  - *(A) Cutting edges incident on a common page can be processed together*
  - *(B) For adjacent partitions, choose the smaller distinct pages incident on the cutting pages to process*
  - *(C) The cutting edges between consecutive partitions in the loading sequence can be saved by reusing pages in memory*



(A)



(B)



(C)

The effect of SGP on the Join Index Graph

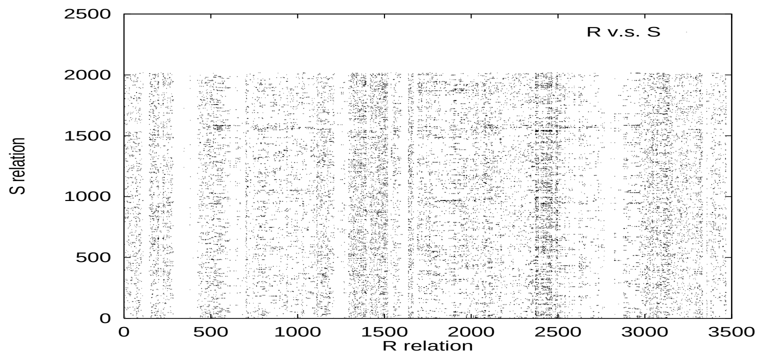


Figure 7: Original PCG

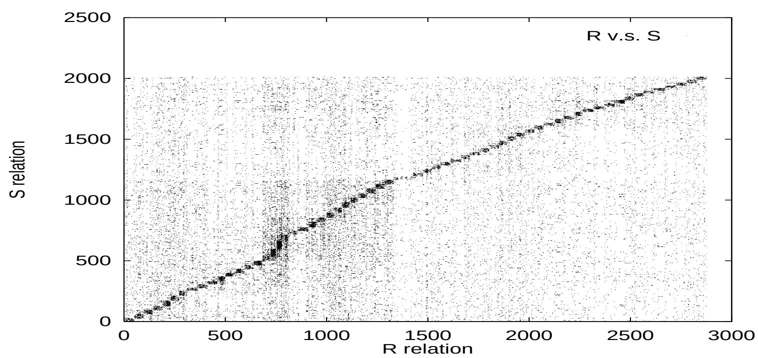


Figure 8: After Partition

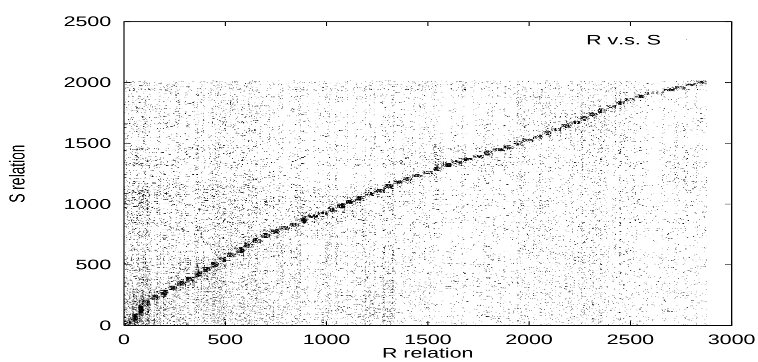
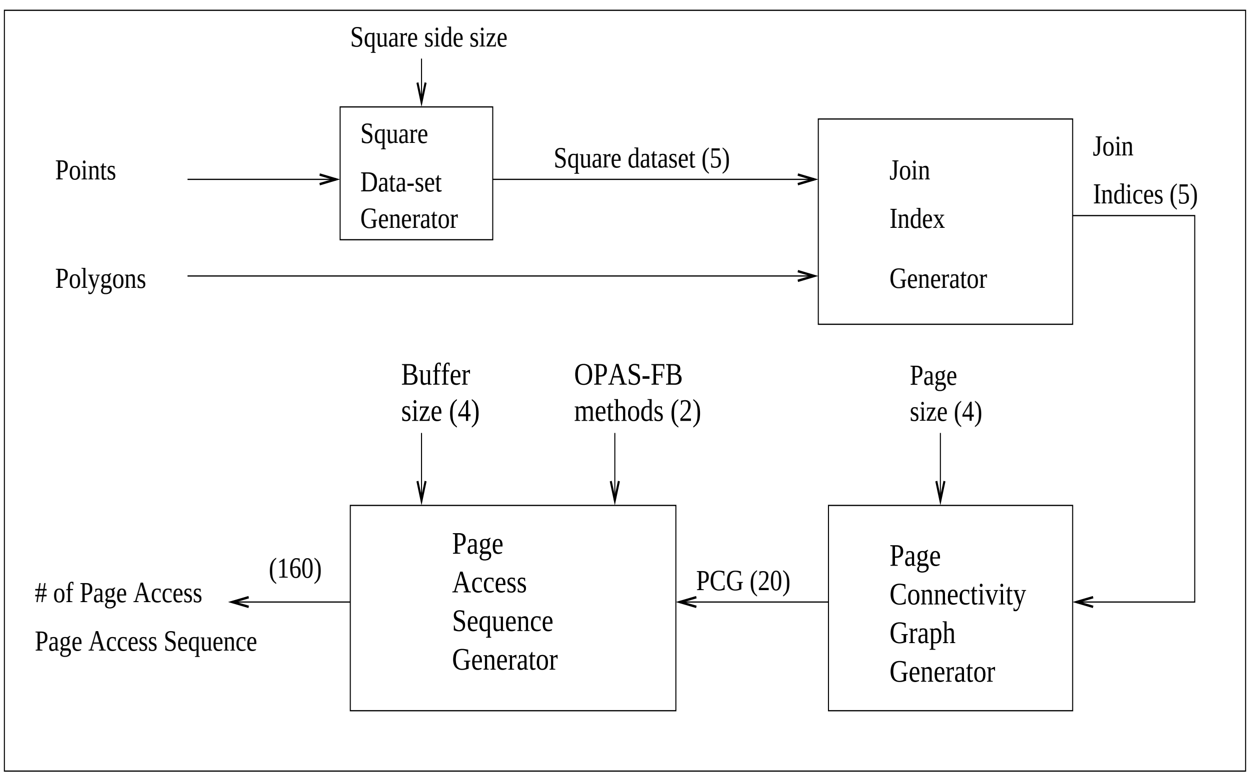


Figure 9: After Partition and Ordering

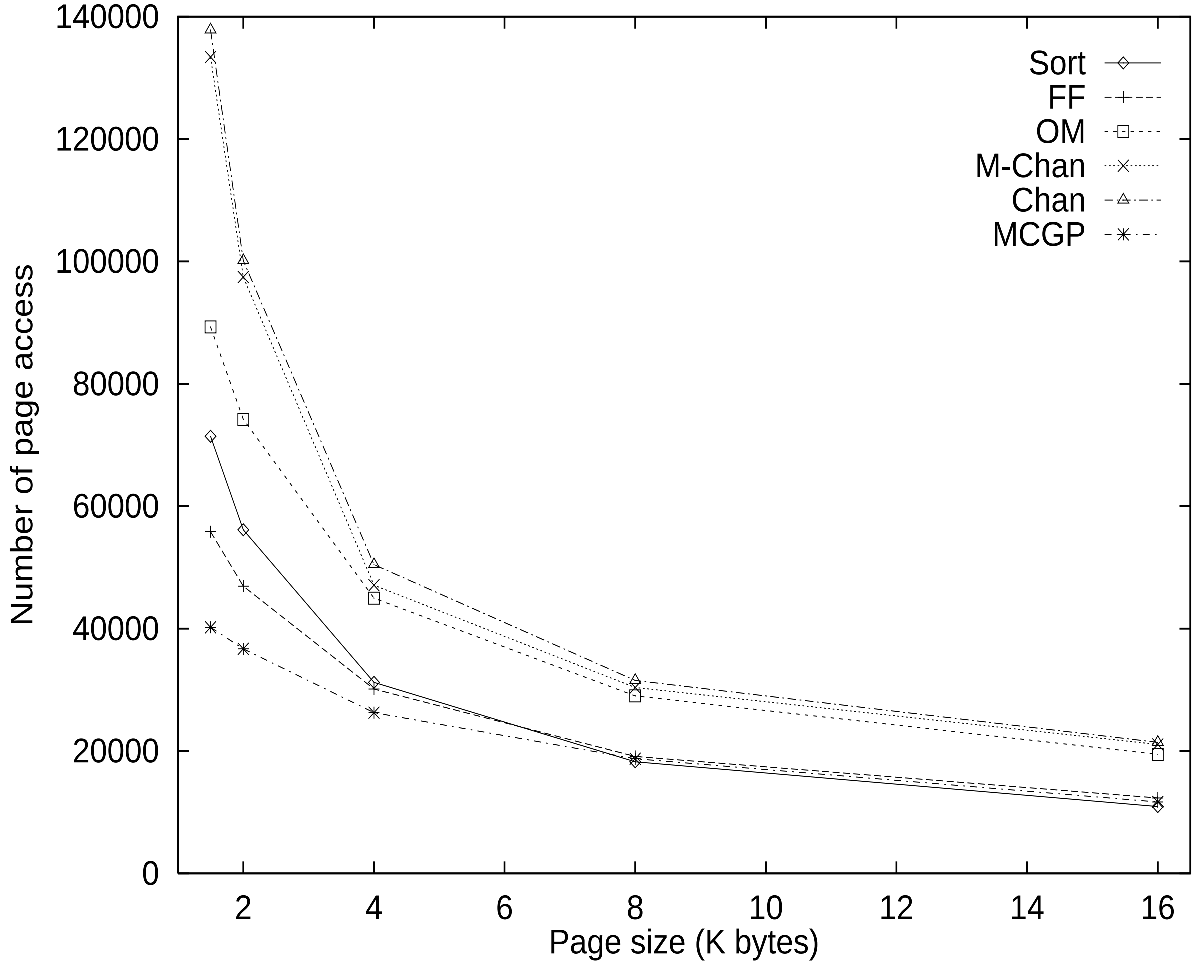
Experiment design

- Methods
  - *Sorting, Fotouhi, Omincinshi, Chan, SGP, AGP*
- Parameters change
  - *Page size, Buffer size, Join Selectivity*
- Data set
  - *Sequoia 2000*
  - *Points: 62,584 California place names + locations*
  - *Polygons: 4388 records, Pasture landuse in CA*



Experiment Result - Effect of page size

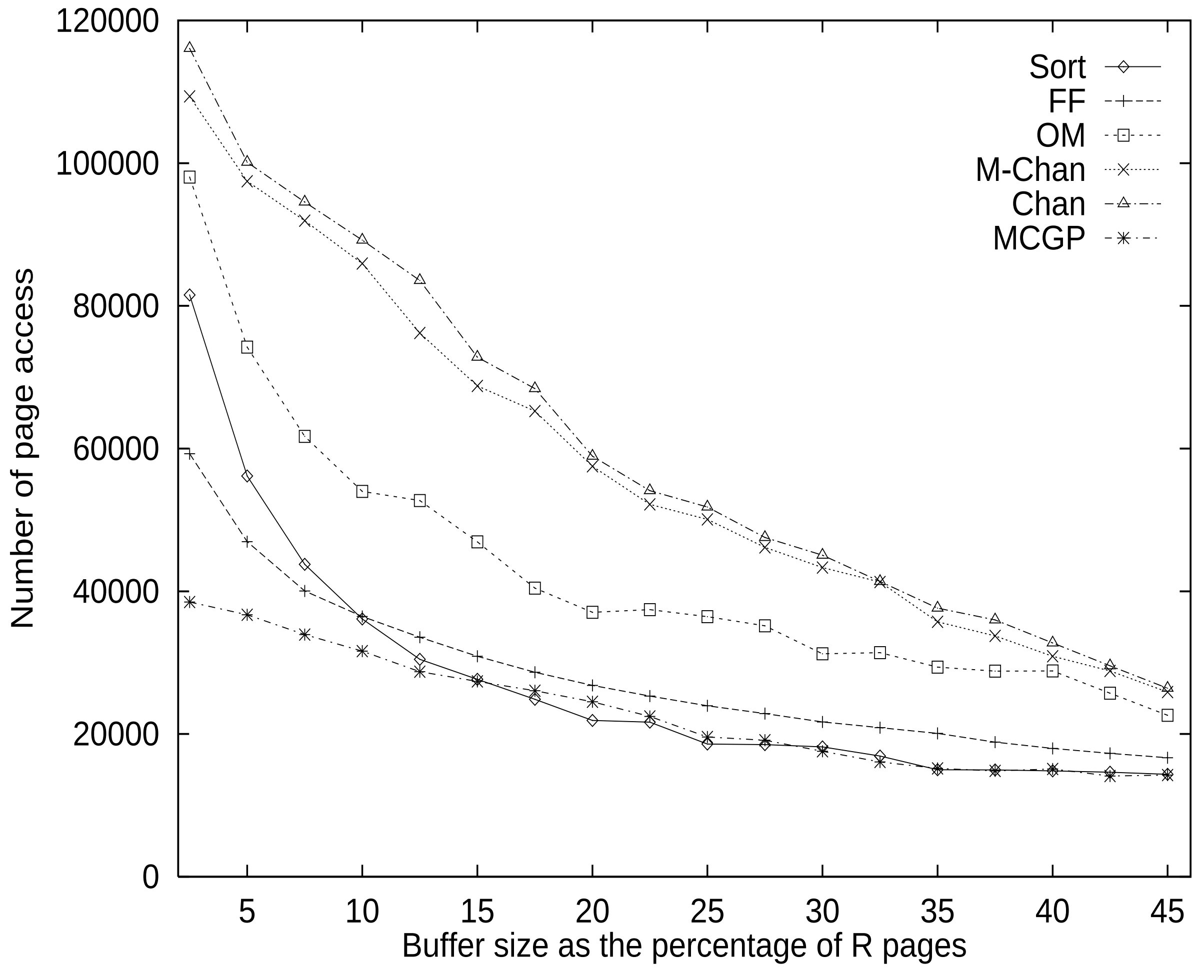
- Fixed Parameters: Buffer size, Join Selectivity
- Variable Parameters
  - *Page size: 2k to 16k*
  - *OPAS-FB heuristics*



- Trend
  - *Page size is small, SGP $\gg$ other methods*
  - *When page size increases, differences decrease*

Experiment Result - Effect of buffer size

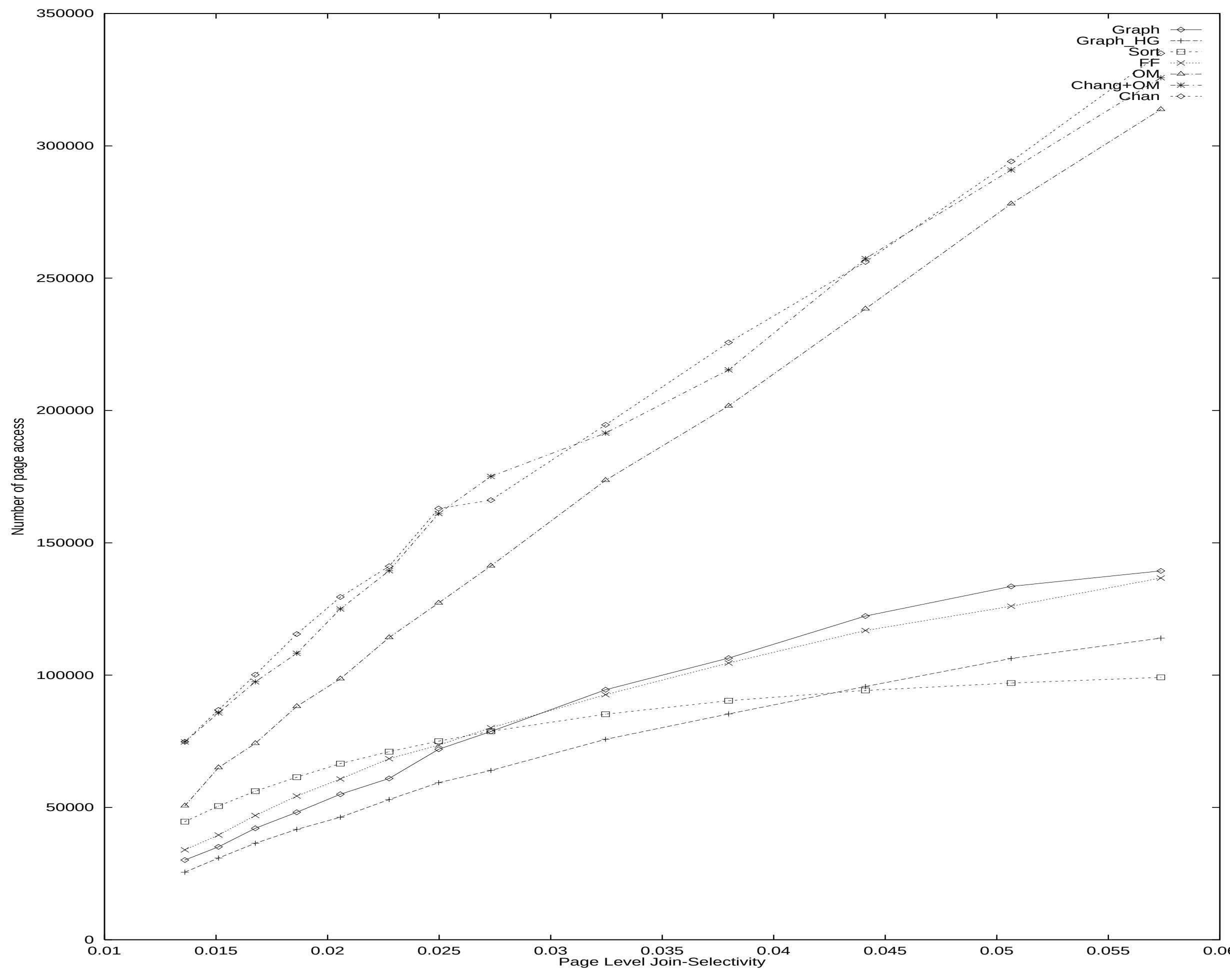
- Fixed Parameters: Page size, Join Selectivity
- Variable Parameters
  - *Buffer size: 5 to 45 percent of the first relation*
  - *OPAS-FB heuristics*



- Trend
  - *Buffer size is small, SGP $\gg$ Other methods*

Experiment Result - Effect of join selectivity

- Fixed Parameters: Page size, Buffer size
- Variable Parameters
  - *Page level join selectivity: 0.01 to 0.06*
  - *OPAS-FB heuristics*



- Trend
  - *Join Selectivity is small, SGP $\gg$ other methods*

Conclusion and Future Work

- AGP and SGP exploits
  - *Global optimization*
- SGP outperforms other methods when
  - *Join Selectivity is low*
  - *Buffer size is "small"*
  - *Page size is small*
- Future work
  - *Multi-way join-indices and hyper-graph partitioning*
  - *Different spatial join predicates*
    - Direction
    - Distance
  - *Join point dataset with stream dataset*

Conclusion and Future Work

R									
7	0	0	0	0	0	0	0	1	
6	0	0	0	0	0	0	1	0	
5	0	0	0	0	0	1	0	0	
4	0	0	0	0	1	0	0	0	
3	0	0	0	1	0	0	0	0	
2	0	0	1	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	7	S

(a)Thickness=1

R									
7	0	0	0	0	0	1	1	1	
6	0	0	0	0	1	1	1	1	
5	0	0	0	1	1	1	1	1	
4	0	0	1	1	1	1	1	0	
3	0	1	1	1	1	1	0	0	
2	1	1	1	1	1	0	0	0	
1	1	1	1	1	0	0	0	0	
0	1	1	1	0	0	0	0	0	
	0	1	2	3	4	5	6	7	S

(b)Thickness=5

R									
7	0	0	0	0	0	1	1	1	
6	0	0	0	0	1	1	1	1	
5	0	0	0	1	1	1	1	1	
4	0	0	1	1	1	1	1	1	
3	0	1	1	1	1	1	1	0	
2	1	1	1	1	1	1	0	0	
1	1	1	1	1	1	0	0	0	
0	1	1	1	1	0	0	0	0	
	0	1	2	3	4	5	6	7	S

(c)Thickness=6

R									
7	0	0	0	0	1	1	1	1	
6	0	0	0	1	1	1	1	1	
5	0	0	1	1	1	1	1	1	
4	0	1	1	1	1	1	1	1	
3	1	1	1	1	1	1	1	0	
2	1	1	1	1	1	1	0	0	
1	1	1	1	1	1	0	0	0	
0	1	1	1	1	0	0	0	0	
	0	1	2	3	4	5	6	7	S

(c)Thickness=7

R									
7	1	1	1	1	1	1	1	1	
6	1	1	1	1	1	1	1	1	
5	1	1	1	1	1	1	1	1	
4	1	1	1	1	1	1	1	1	
3	1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	
0	1	1	1	1	1	1	1	1	
	0	1	2	3	4	5	6	7	S

(c)Thickness=15

Figure 10: Table for different edge ratio

Class	1:1	M:N	M:N	M:N	M:N
Edge Ratio	8/64	34/64	39/64	44/64	1
AGP/Sorting	16	20	21	22	24
OM	16	16	18	21	35
FP	16	16	23	26	32
Chan	16	16	18	23	35
SGP	16	16	21	26	34

Table 3: I/O count for different methods

<table>
<tr><td>Domain</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>One-dimension</td><td>No method has redundant Cheapest=Sorting</td><td rowspan="2"></td><td rowspan="2">No symmetric method has redundant I/O Cheapest=FP</td><td rowspan="2"></td><td rowspan="2">Small vertex cover for off B diagnal edges $\Rightarrow$ SGP,

Some cases, OM,FP, or Chan may be slightly better.</td><td></td><td>Asymmetric methods have less redundant I/O. Cheapest = Sorting.</td></tr>
<tr><td>Multi-dimension</td><td>No method but sorting has redundant I/O Cheapest = AGP</td><td></td><td>AGP has least I/O</td></tr>
<tr><td>Relationship Cardinality</td><td>1:1
1:N</td><td colspan="6">M:N</td></tr>
<tr><td>Tinckness of diagnonal</td><td>1</td><td colspan="3">$\leq B$</td><td colspan="2">$> B$, but few entries outside B diagonal</td><td>Fully connected</td></tr>
</table>

Table 4: Summary