A Partitioning Algorithm for Block-Diagonal Matrices with Overlap

Guy Atenekeng [1]
Laura Grigori[2] Masha Sosonkina[3]

[1]IRISA

[2]INRIA Futurs

[3]Ames Laboratory, Iowa State University

Parallel Matrix Algorithm and Application, 2006

Outline

1. Problem and motivations
   - Standard definition of graph partitioning
   - Our partitioning problem
2. First step of our algorithm: obtain an initial partition
   - Reduce the band of the input matrix
3. Second step of our algorithm: refinement of the initial partition
   - Scheme to decrease the size of the separators
4. Experimental results
5. Conclusion

Standard definition of graph partitioning

Definition

Let $G = (V, E)$ be the adjacent graph of a given matrix A. A decomposition of V into K disjoint subsets $V_1, V_2, \cdots, V_k$, such that $\cup_i V_i = V$ is called a $K - way$ partitioning of vertex set V.

Definition

- A $K - way$ partitioning of V satisfies a load balance constraint specified by $[l, u]$, if for each part V_i, $l <| V_i |< u$.
- The cut of a $K - way$ partitioning of V is equal to the number of edges that contain vertices from different parts V_i.

Standard definition of graph partitioning

Characteristics

- Graph partitioning is an NP hard problem.
- Graph partitioning is a combinatorial optimization problem.
- Generally, the optimization function consists of minimizing the cut while maintaining some constraint.

Solution

- Many algorithms have been developed which produce reasonably good partitionings: spectral partitioning, geometric partitioning, multilevel graph partitioning, etc.
- Several libraries exist for graph partitioning: METIS, PATOH, SCOTCH, etc.

Our partitioning problem

Our graph partitioning problem has additional constraints.

Definition

A two-neighboring graph partitioning of G into K partitions is defined by the sets of vertices $V_i \subset V$, $i = 1, ..., K$ such that if $|i - j| > 1$ then there is no edge between V_i and V_j.

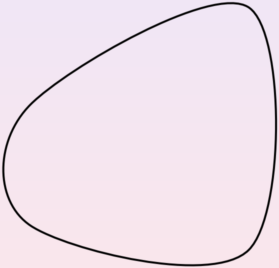
Consequence

Consequence of our definition

1. Each partition has at most 2 neighbors.
2. The permutation obtained by this partition leads to a block diagonal form.
3. The set of vertices that share an edge between two partitions forms an overlap between the two partitions.

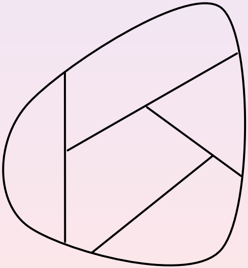
This form of block diagonal with overlaps is suitable for an explicit formulation of the multiplicative Schwarz preconditionner [Atenekeng, Kamgnia, Philippe'05].

Illustration



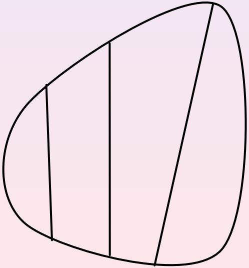
Original Domain

Illustration



Standard Partitioning

Illustration



Our partitioning

Our general scheme for obtaining block-diagonal matrices with overlap

Algorithm

1. Apply a permutation to reduce the band of the input matrix.
2. Form a spanning tree of the permuted matrix.
3. Find an initial partition from the spanning tree.
4. Refine the separators obtained at 3.

Reduce the band of the input matrix

Algorithm

$B = A + A^T$ or $B = AA^T$
$p = RCM(B)$
$B = B(p, p)$
$T = BFS(B, 1)$
RCM: Reverse Cuthill Mckee
BFS: Breadth-first search

Remark

We choose 1 as the starting point in the Breadth-first search because $p(1)$ is a peripheral point of the input matrix.

Partitioning of the spanning tree

We use an algorithm that partitions a chain of tasks. [Pinar, Aykanat' 04].
For this, we need to define the ”Tasks” and the ”Weights”.

Definition

- A task is defined as the union of two consecutive levels in the spanning tree. The number of tasks is $N/2$.
- The weight of each task can be either the number of vertices in a task or the sum of the degree of vertices in the task.

Partitioning of the spanning Tree

Chains on Chains partitioning (CCP)

- The objective of the CCP problem is to find a sequence $\Pi_K =< s_1, s_2, ..., s_K >$ of $K - 1$ separators to divide a chain of N tasks into K consecutive parts.
- This division is obtained by minimizing the maximum of load per processor.

For details see [Pinar, Aykanat' 04]

An initial partition is obtained from the spanning tree and the index sequence Π_K.

Description

1. Compute the index separator of the spanning tree. $u_p = 2 * s_p$
2. Vertices in level $u_{p-1} + 1$ to u_p form the partition p.
3. An initial vertex separator is obtained by finding the minimum vertex cover between levels u_{p-1} and $u_{p-1} + 1$.

Remark

Let N be the number of levels in the spanning tree.

- The number of levels in the spanning tree T influences the quality of the partitioning: the more levels, the better the partitioning.
- If $N < K$ then it is not possible to partition the vertices of the graph G into K partitions.

General scheme of refinement

We denote by S_i the separator between partitions Ω_i and Ω_{i+1}.

Graph: G

We can represent our graph now as follow:

$$G = [\Omega_0, S_0, \Omega_1, ..., \Omega_{p-1}, S_{p-1}, \Omega_p] \quad (1)$$

The main objective now is to refine a separator while maintaining this structure and a load balancing constraint.

General scheme of refinement

Remark

There are two raisons why we are more interested in vertex refinement than edge refinement:

1. The number of iteration to convergence in Krylov subspace method preconditioned by *EFMS* is limited by $| \cup_{i=0}^{p-1} S_i |$.
2. $| \cup_{i=0}^{p-1} S_i |$ represents the total size of exchange data in matrix vector product operation.

General scheme of refinement

Algorithm

1. Choose a separator S_j to improve
2. Improve S_j
3. Return to step 1 until further improvement is not possible .

The separator S_j is chosen such that:
$S_j = max\{S_i, i = 0, ..., p-1\}$

Improved S_j with Ω_j and Ω_{j+1}

The improvement consists of finding $Y \subset S_j$ such that $\mid Adj(Y, \Omega_j) \mid < \mid Y \mid$ or $\mid Adj(Y, \Omega_{j+1}) \mid < \mid Y \mid$.

Algorithm

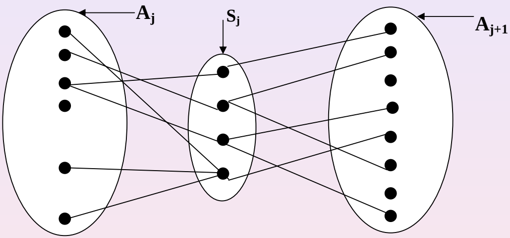
1. We compute Y in bipartite graph define by S_j and Ω_j by using a standard augmenting path algorithm. For details, see [Liu' 1989]
2. Now Y has been found, update Ω_{j+1} and Ω_j.
3. Return to step 1 until no further improvement is required.

Update partition

Update partition

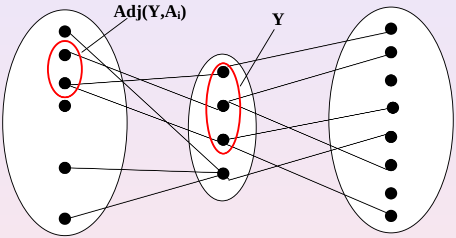
1. $\Omega_{j+1} = \Omega_{j+1} \cup Y;$
2. $S_j = (S_j - Y) \cup Adj(Y, \Omega_j);$
3. $\Omega_j = \Omega_j - Adj(Y, \Omega_j)$

Illustration: Update partition (standard case)



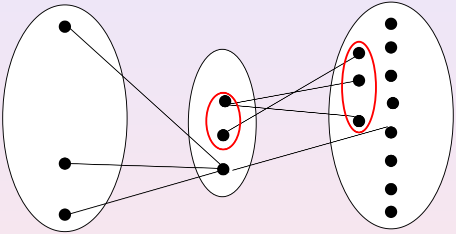
Improvement Separator

Illustration: Update partition (standard case)



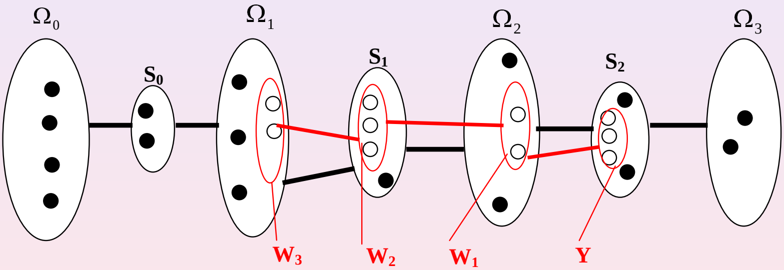
Find subset Y

Illustration: Update partition (standard case)



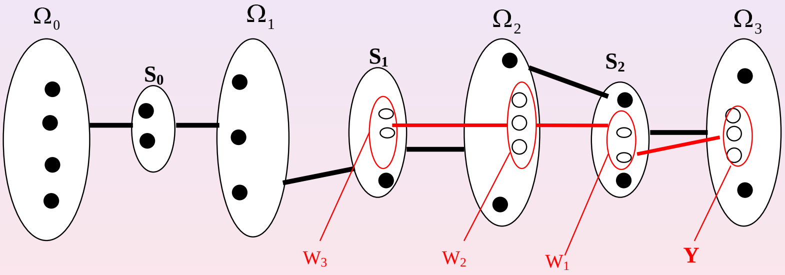
Update

Illustration: Update partition (our case)



Initial position of moveables vertices

Illustration: Update partition (our case)



Graph after vertices moves

Experimental results

- We use sparse matrices from collection of University of Florida.

MATRIX	N	NNZ	MD	AD	NUML	MAXCL	AVGL
scircuit	170998	958936	5	353	202	7533	846

NP	MAX/AVG	HM MAX/AVG	SEP SIZE	SEP SIZE/N
4	1.21	1.02	16744	0.09
8	2.05	1.02	37504	0.21
16	2.66	1.05	69899	0.40

MD: Maximum degree of a vertex in the graph of A.
AD: Average degree of a vertex in the graph of A.
NUML: Number of levels in the spanning tree.
MAXCL: Maximum cardinality of levels in the spanning tree.
AVGL: Average cardinality of levels in the spanning tree.
MAX/AVG : Maximum partition size / Average partition size
HM MAX/AVG: Maximum partition size / Average partition size obtained by HMETIS.
SEP SIZE: Sum of separators size.

SEP SIZE/N: Sum of separators size/Input matrix size

Experimental results

MATRIX	N	NNZ	MD	AD	NUML	MAXCL	AVGL
torso3	259156	4429042	23	17	107	7750	2422

NP	MAX/AVG	HM MAX/AVG	SEP SIZE	SEP SIZE/N
4	1.06	1.03	11252	0.04
8	1.20	1.05	29376	0.11
16	1.25	1.01	52987	0.2
32	1.63	1.05	100568	0.38

MATRIX	N	NNZ	MD	AD	NUML	MAXCL	AVGL
Stomach	213360	3021648	22	16	379	1124	562

NP	MAX/AVG	HM MAX/AVG	SEP SIZE	SEP SIZE/N
4	1.01	1.02	1680	0.007
8	1.02	1.02	4373	0.02
16	1.05	1.03	8958	0.04
32	1.17	1.05	18127	0.08

MATRIX	N	NNZ	MD	AD	NUML	MAXCL	AVGL
af_shell9	504855	9046850	34	40	486	2605	1038

NP	MAX/AVG	HM MAX/AVG	SEP SIZE	SEP SIZE/N
8	1.05	1.01	11391	0.02
16	1.11	1.02	23432	0.04
32	1.17	1.02	48455	0.09
64	1.54	1.05	103169	0.2

Conclusion

Conclusion: Improvement

- The time of the refinement is critical. As future work, we need to decrease the time of the refinement step.
- The quality of partition is strongly depend to quality of spanning tree.

Complexity

Let N and NNZ be the size of given matrix.
Let N_L be the number of level in spanning tree.
Let $|S|$ be the cardinality of separator and $|E_M|$ be the maximum number of edges between partitions and separators.

Complexity

- $O(K(N_l - k) * log(k) + 2 * NNZ + N + |S| * |E_M|)$