

# Improved Data Partitioning by Nested Dissection with Applications to Information Retrieval

Michael M. Wolf\*      Erik G. Boman†      Cédric Chevalier‡

The computational work in many information retrieval and analysis algorithms is based on sparse linear algebra. Sparse matrix-vector multiplication is a common kernel in many of these computations. Thus, an important related combinatorial problem in parallel computing is how to distribute the matrix and the vectors among processors so as to minimize the communication cost. We focus on minimizing the total communication volume while keeping the computation balanced across processes. In [1], the first two authors presented a new 2D partitioning method, the *nested dissection partitioning algorithm*. In this paper, we improve on that algorithm and show that it is a good option for data partitioning in information retrieval. We also show partitioning time can be substantially reduced by using the SCOTCH software, and quality improves in some cases, too.

## One- and Two-Dimensional Partitioning

In 1D sparse matrix partitioning, each part is assigned the nonzeros belonging to a set of rows or a set of columns. 1D partitioning is sufficient for many problems, and most applications use matrices distributed in a 1D manner. However, for particular problems, 1D partitioning is potentially disastrous in terms of the communication volume. Thus, we need more flexible partitioning than traditional 1D partitioning. 2D partitioning is a more flexible alternative to 1D partitioning, in which parts are assigned more general sets of nonzeros. This problem can be modeled as a *fine-grain* hypergraph[2], where each nonzero is represented by a vertex in the hypergraph, allowing each nonzero to be assigned a part independently. Catalyurek and Aykanat [2] proved that this fine-grain hypergraph model yields a minimum volume partitioning when optimally solved. A drawback of the fine-grain hypergraph partitioning problem is that it is a larger NP-hard problem (than 1D partitioning) and thus, may be too expensive to solve for large matrices.

## Nested Dissection Partitioning Algorithm

In [1], we proposed a new 2D method, the nested dissection partitioning algorithm, for structurally symmetric matrices. This algorithm can be generalized and applied to nonsymmetric matrices by the means of a bipartite graph model [1]. In [1], we presented an accurate graph model for communication volume in matrix-vector multiplication. We partition both the vertices and edges, which distinguishes our approach from the 1D graph model and allows for 2D partitioning. If we solved this exact graph model optimally, we would obtain a balanced partition to minimize communication volume for resulting matrix-vector multiplication.

Our nested dissection partitioning algorithm solves this exact graph model suboptimally but in polynomial time (the problem is NP-hard). For simplicity, we consider bisection first. First we compute a small balanced vertex separator  $S$  for the graph. This partitions the vertices into three disjoint subsets  $(V_0, V_1, S)$ . Let  $E_j := \{e \in E | e \cap V_j \neq \emptyset\}$  for  $j = 0, 1$ .  $V_j$  and  $E_j$  are assigned to part  $P_j$  for  $j = 0, 1$ . In practice, one wishes to partition into  $k > 2$  parts. We compute a  $k$ -separator via recursive bisection (“nested dissection”) and apply the above procedure recursively to the subdomains. The steps of our vertex separator graph partitioning algorithm are given in Algorithm 1. The calculation of the vertex separators (line 1) gives us  $k$  disjoint subdomains divided by a hierarchy of  $k - 1$  separators. The algorithm does not depend on any particular method for calculating the vertex separators, but smaller separators tend to yield lower communication volumes. Lines 2 and 3 are fully expounded in Algorithm 1. However, there are many different ways to distribute the separator vertices and edges connecting separator vertices (lines 4-6), yielding several variations on our basic algorithm. In our initial implementation, we assigned the separator vertices and edges to parts using a heuristic that allowed us to bound the communication volume. We showed in our previous work [1] that this initial implementation produced fairly good results, similar quality to the fine-grain method but requiring less runtime. However, we believed that further improvements could be made by focusing on the separator partitioning.

We developed a model for improving the partitioning of the separator vertices and edges connecting separator vertices (lines 4-6 in Algorithm 1). We focus only on partitioning these separator vertices and edges, given the previous partitioning of the rest of the graph. We replace the interior subdomain vertices with special fixed vertices, one for each part of the partition. The fixed vertices serve as a mechanism to account for the resulting

---

\*CS Dept., Univ. of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. (mmwolf@uiuc.edu).

†Scalable Algorithms Dept., Sandia National Laboratories, Albuquerque, NM, USA. ({egboman,ccheval}@sandia.gov).

---

**Algorithm 1** Nested Dissection Graph Partitioning

---

- 1: Compute  $k - 1$  small vertex separators.  $\triangleright k$  is the desired number of parts.
  - 2: Assign part  $P_i$  to vertices in  $V_i$ .  $\triangleright V_i$  is set of vertices in the subdomain  $i$ .
  - 3: Assign  $P_i$  to edges in  $E_i$ .  $\triangleright E_i$  is set of edges that contain a vertex in  $V_i$ .
  - 4: Assign parts to separator vertices.
  - 5: Assign parts to edges that connect separator vertices.
- 

communication when a separator vertex is assigned a part different from the part of a neighboring interior vertex. We partition the nonzeros corresponding to the separator vertices and edges by using a symmetric version of the fine-grain hypergraph method. We can use the relatively slow fine-grain hypergraph method here without jeopardizing total run-time since we only partition a small fraction of the original matrix. Our goal is to obtain a partitioning of the separator vertices and edges that is superior to that obtained by our initial implementation.

## Results

In our work, we compared the partitionings of different methods for a set of nine sparse information retrieval matrices. We studied term-document, term-term, and web link matrices. We compare the original nested dissection algorithm and the new two-phase algorithm with the 1D (row and column) hypergraph methods and the fine-grain hypergraph method. We also tested two different approaches to compute vertex separators in our two-phase algorithm. First, we derive our vertex separators from edge separators produced by hypergraph partitioning. We also implemented a version that uses SCOTCH to compute the vertex separators. We partitioned the matrices into  $k = 2, 4, 16, 64, 256$  parts to study the behavior of the algorithms with varying  $k$ .

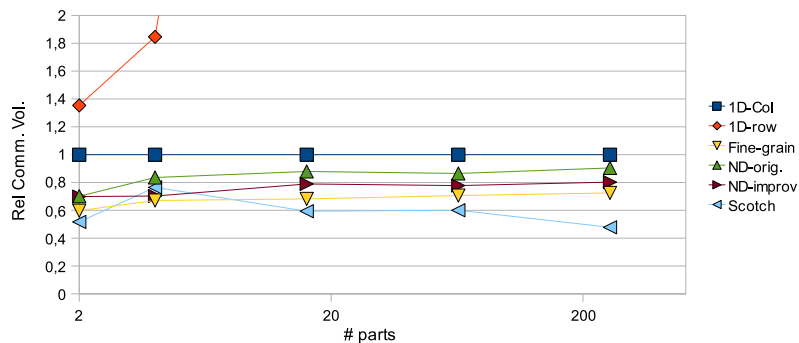


Figure 1: Average communication volume for web link matrices, normalized to 1 for 1D column partitioning results. For  $k > 4$ , 1D row partitioning results are too large to fit in plot.

For these information retrieval matrices, we saw a reduction in communication volume for our improved implementation over the original implementation. For most of the matrices, our improved nested dissection algorithm yields significantly lower volume partitions than the 1D methods and is competitive with the fine-grain method. Figure 1 show the average communication volumes obtained by partitioning the web link matrices for the different partitioning methods. We see that the fine-grain method produced partitions of significantly less volume than the original nested dissection method and slightly less than the improved two-phase method. However, the SCOTCH based nested dissection method was superior to the fine-grain and other methods for these web link matrices. In practice, a matrix is perhaps used for only a few hundred iterations so the partitioning time is also important. Timings for these methods showed extremely large relative runtimes for the fine-grain hypergraph method. In general, the first two nested dissection methods that use 1D partitioning to find the separators are significantly faster than the fine-grain method. The nested dissection based method that uses SCOTCH to find the separators is typically the fastest method, even faster than the 1D hypergraph methods. This is expected since graph partitioning is faster than hypergraph partitioning.

## References

- [1] E. G. Boman and M. M. Wolf. A nested dissection approach to sparse matrix partitioning for parallel computations. Technical report, Sandia National Laboratories, NM, 2008. Submitted.
- [2] Ü. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2d decomposition of sparse matrices. In *Proc. IPDPS 8th Int'l Workshop on Solving Irregularly Structured Problems in Parallel (Irregular 2001)*, April 2001.