

Homework #3, AA220 / CME 342 / CS238, Spring, 2005.

Assigned: Wednesday 5/11/05. Due: Wednesday 5/18/05.

GRAPH PARTITIONING FOR PARALLEL COMPUTING ON UNSTRUCTURED MESHES

In order to handle complex geometries in the solution of PDE-governed physical systems, a popular approach is to discretize the space of interest using an unstructured mesh with various kinds of cell types (tetrahedra, prisms, pyramids, hexahedra, etc.) that can overcome typical topology limitations of structured and block-structured meshes. This approach is typical in Structures, Computational Fluid Dynamics, Heat Transfer, and Electromagnetics calculations.

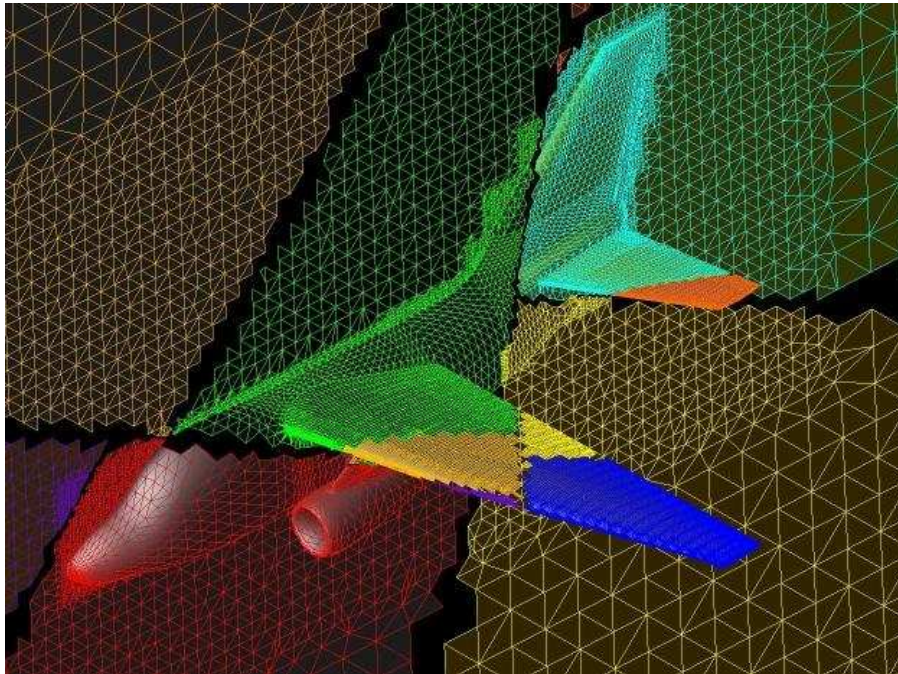


Figure 1: Result of Graph Partitioning for S3-A Aircraft

An unstructured mesh can be represented by a weighted, undirected graph, $G = (N, E, W_N, W_E)$ such as the ones we have been discussing in class. In this case, the weights for all nodes and edges are taken to be $W_N, W_E = 1$ since each node in the graph has to do approximately the same amount of work (except perhaps the boundary nodes) and each edge represents a transfer of information which is about equal (with slight variations as well).

An often-used procedure to implement the solution of this problems in parallel computers is to partition the graph into a number N_p of partitions N_i such that $N = N_1 \cup N_2 \cup \dots \cup N_p$ that represent the given mesh and that

1. Distribute the sum of the nodal weights, W_N within each partition of the graph as evenly as possible among all N_p processors (every processor does almost the same amount of work - *load balancing*.)
2. Minimize the sum of the weights of the edges, W_E that communicate the various partitions $N = N_1 \cup N_2 \cup \dots \cup N_p$. This minimizes the amount of data to be communicated between different processors, thus decreasing total parallel overhead.

A sample surface decomposition of a volume mesh for an aircraft calculation can be seen above.

Assignment:

- (a) Write a program that initializes a graph using the data file format provided and that performs a geometric partitioning that attempts to construct partitions with as close to an even number of nodes as possible. Pay no attention to the communication requirements mentioned above.
- (b) Modify your program to use the Metis library (installed on *junior.stanford.edu* and downloadable from <http://www-users.cs.umn.edu/karypis/metis/main.shtml>). Please attempt to install the Metis program yourselves on a UNIX/Linux machine of your choice to gain some experience in these matters.
- (c) Run both programs for two of the three unstructured meshes provided (varying sizes) and for $N_p = 2, 4, 8, 16$, and report on your findings regarding load balancing issues, total communication costs, CPU time.

Final Project Description:

With this assignment, I would like you to turn in a brief (< 1 page) description of the final project that you intend to tackle for AA220/CME342/CS238. The expected work for this final project should be equivalent to two homework assignments.

The intent is to provide you with some feedback as to the suitability and feasibility of such a project. Ideally, the project would be related to work that you are doing as part of your research.

If you are not currently doing work that may require parallel methods, I will be assigning a 5th and final homework that you can use in lieu of the final project. If this is the case, you do not need to provide us with a description of the the final project.

Optional Assignment...with prize!

This portion of the assignment is optional. No extra credit will be given. If you do come up with the best implementation of the partitioning for these graphs you will win a lunch for 2 at the Faculty Club. That, together with the fact that the winner will be announced on the class wiki (wow!) is the only motivation.

The Metis partitioner fails to produce adequate partitions when the number of nodes assigned to a given processor is small. In the TFLO2000 solver that we use in our ASC program, the graph to be partitioned has nodes that represent entire blocks in a multiblock-structured mesh and edges that represent the required communication between blocks in the mesh. The parallelization requires that entire blocks be assigned to a given processor. For very large numbers of processors,

each processor typically only deals with the solution in one or two of the blocks in the mesh. Metis often leads to partitions with no blocks and the solver, TFLO2000, fails.

On the web page you are provided with 3 graphs that represent the multiblock mesh around a Pratt & Whitney 6000 engine turbine. The following FORTRAN code shows the information contained in these files.

```
read(20,*)
read(20,*) nVertex, nCon

read(20,*)
do ii=0,nVertex
  read(20,*) xadj(ii)
enddo

read(20,*)
do ii=1, xadj(nVertex)
  read(20,*) adjncy(ii), adjwgt(ii)
enddo

read(20,*)
do ii=1,nVertex
  read(20,*) vwgt(1,ii), vwgt(2,ii)
enddo
```

where, `nVertex` is the number of blocks to be partitioned (vertices of the graph), `nCon` is the number of constraints used to partition the graph (Metis typically can use a multi-constrained partitioning scheme. This will be 2, namely number of cells and number of faces of blocks that need to be communicated). `xadj` is the number of edges per vertex in cumulative storage format. `adjncy` is the end vertex of each of the edges that start at one vertex; the size of `adjncy` is `xadj(nVertex)`. `adjwgt` is the edge weight; the number of first level halos to be communicated between the blocks/vertices; also this array has size `xadj(nVertex)`. `vwgt` are the vertex weights = number of cells and number of faces in a block.

C-numbering is used, i.e. everything starts at 0.

The assignment is to come up with the best partitioning of these graphs, using a single algorithm. More details on how the “best” partitioning will be provided on the class wiki page.