Optimization of Mesh Partitions and Impact on Parallel CFD

D. Vanderstraeten\textsuperscript{a*}, R. Keunings\textsuperscript{a}, and C. Farhat\textsuperscript{b}

\textsuperscript{a}Centre for Systems Engineering and Applied Mechanics, Université Catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

\textsuperscript{b}Department of Aerospace Engineering Science and Center for Space Structures and Controls, University of Colorado at Boulder, Boulder, CO 80309-0429, U.S.A.

Most of the recently proposed parallel CFD algorithms stem from the "divide and conquer" paradigm and involve some form of domain decomposition. When the discretization is highly regular and when the number of desired subdomains is such that a regular mesh decomposition is possible (i.e. box or strip decompositions), finding an appropriate mesh partition is a trivial task. However, the problem becomes more challenging when dealing with unstructured meshes. Here, we explore a two-step procedure for partitioning a given unstructured mesh and attaining a specific objective. First, we construct an initial partition using a fast deterministic algorithm. Next, we refine the initial decomposition with a non-deterministic optimization algorithm in order to reach a specific goal. This method has proven to produce decompositions with close-to-optimal interface sizes and with a relatively low CPU overhead.

1. INTRODUCTION

The automatic partitioning of unstructured grids is currently the topic of much research effort. The main motivation for these developments is the efficient numerical solution of partial differential equations by means of domain decomposition techniques. In this paper, we consider a domain decomposition approach where the grid is decomposed geometrically into a predefined number of non-overlapping domains.

The problem consists of distributing the elements of an unstructured mesh among a specified number of subdomains. The partitions should (a) reduce interprocessor communication costs to their minimum, and (b) ensure an equal compute load among the processors.

We have developed a two step approach in order to reach these goals. First, we decompose the grid with a fast deterministic algorithm. Second, a more complex optimization is applied to the initial partition in order to achieve a better load balance and to reduce the interface size as much as possible. Three different optimization algorithms have been investigated, namely Simulated Annealing, Stochastic Evolution, and Tabu Search.

In the next section, we briefly mention some of the available direct decomposition

\textsuperscript{*The Ph.D. work of Denis Vanderstraeten is supported by the Fonds National de la Recherche Scientifique of the Belgian State.}
algorithms. Next, we describe the optimization algorithms used in this work. Finally, we report performance results that highlight the superiority of the proposed two-step approach over conventional one-step algorithms.

2. INITIAL DECOMPOSITION

A large set of direct algorithms has been described in the literature. Eleven techniques have been tested in the present work.

One of the most popular algorithms is the Recursive Spectral Bisection (RSB). It creates a bipartition by using the second eigenvector associated to the Laplacian matrix of the dual grid [1]. Another well-known technique is the Greedy algorithm (G) that uses local information on the grid [2,3]. We also investigate a slightly modified version of the Recursive Graph Bisection (RGB) that handles better unstructured grids [4]. Finally, we have implemented the Reverse Cuthill-Mc-Kee algorithm (RCM), three inertial algorithms along the three principal axes (PIX, PIY, PIZ) and their recursive variants (RRCM, RPI) [3].

3. OPTIMIZATION WITH NON-DETERMINISTIC ALGORITHMS

First, we introduce a cost function to define the decomposition requirements. For example, if interprocessor communication is to be minimized and load balance is to be achieved among the processors, the cost function can be written as:

\[
\text{Cost} = \alpha \left| \left\{ (i,j) \mid (i,j) \in E' \text{ and } \text{Subdomain}(i) \neq \text{Subdomain}(j) \right\} \right| + (1 - \alpha) \sum_{k=1}^{N_s} (N_k - N_{\text{best},k})^2,
\]

where \( N_s \) denotes the number of subdomains. By letting \( N_k \) denote either the number of elements, or the number of nodes, or the number of edges in subdomain \( k \), \( \text{Cost} \) attempts at balancing the element-wise, node-wise, or edge-wise parallel computations. \( N_{\text{best},k} \) denotes the optimal number of elements, nodes or edges in subdomain \( k \). Let \( N \) be the total number of elements. If the computational load of the processors is proportional to the number of elements, the value of \( N_{\text{best},k} \) equals \( N/N_s \) for all \( k \). In more complex situations, we do not always know its optimal value before the optimization step; we then use an adaptative scheme that refines the \( N_{\text{best},k} \)'s during the optimization process. The constant \( \alpha \) is a penalty parameter which can be used to favor communication costs over load balancing, or vice-versa.

Given the above cost function, the partitioning problem is formulated as a combinatorial optimization task defined on the dual graph \( G' = (V', E') \) associated with the mesh [4,5]. This problem is known to be \( NP \)-hard. Therefore, we have explored three non-deterministic iterative optimization algorithms that improve initial decompositions produced by different direct techniques in a reasonable computation time. More details about the three heuristics, Simulated Annealing (SA), Tabu Search (TS) and Stochastic Evolution (SE), can be found in the literature [6–10]. The overall procedure goes as follows. A feasible solution is perturbed (i.e. interface elements are moved from one
subdomain to another), and the move is accepted on the basis of a test specific to each heuristic. The interface elements are chosen randomly for SA and for TS, and in a pre-defined order for SE. Moves that reduce the cost function are always accepted, while moves that increase the cost function are accepted with a heuristic dependent probability. Briefly said, this probability depends on the difference of cost before and after a transfer, and decreases with the number of perturbations already performed.

Since the perturbations are located at the subdomain interfaces and since the heuristics start with a good initial solution, the optimization process quickly reaches quasi-optimal solutions.

4. RESULTS

The proposed two-step approach to the partitioning problem has been evaluated on a variety of unstructured finite element meshes.

Representative results are reported for a set of 7 meshes, with a number of elements increasing between 664 and 15998. Decompositions have been generated for a number of subdomains ranging between 2 and 64.

In this section, we first compare the quality of the decompositions obtained by the direct methods. We then discuss the improvement afforded by the optimization step, as well as its added CPU cost. Partitions are evaluated in terms of interface size.

4.1. Direct Decomposition

In Table 1, we show the best direct algorithm in terms of interface nodes. All the algorithms create partitions with an equal number of elements per subdomains. The best decomposition is created 19 times by the RSB and 10 times by the Greedy algorithm. We should note that the quality of the RSB and Greedy methods is similar since the difference in terms of interface nodes is very low. The other decomposers are less attractive.

<table>
<thead>
<tr>
<th>$N_s$</th>
<th>664</th>
<th>1129</th>
<th>3648</th>
<th>4008</th>
<th>4032</th>
<th>7552</th>
<th>15998</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RSB</td>
<td>PIX</td>
<td>RCM</td>
<td>PIX</td>
<td>RSB</td>
<td>RGB</td>
<td>PIX</td>
</tr>
<tr>
<td>4</td>
<td>RSB</td>
<td>RSB</td>
<td>RRCM</td>
<td>PIX</td>
<td>RGB</td>
<td>RSB</td>
<td>PIX</td>
</tr>
<tr>
<td>8</td>
<td>RSB</td>
<td>RSB</td>
<td>RGB</td>
<td>RSB</td>
<td>RGB</td>
<td>RSB</td>
<td>RSB</td>
</tr>
<tr>
<td>16</td>
<td>RSB</td>
<td>RSB</td>
<td>G</td>
<td>RSB</td>
<td>RGB</td>
<td>RSB</td>
<td>G</td>
</tr>
<tr>
<td>32</td>
<td>RSB</td>
<td>RSB</td>
<td>G</td>
<td>RSB</td>
<td>RGB</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>64</td>
<td>RSB</td>
<td>G</td>
<td>G</td>
<td>RSB</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

Table 1. Best direct algorithm in terms of interface size. The name in a cell is the name of the algorithm that produces the decomposition with the lowest number of interface nodes.

Figure 1 shows the CPU time on a SGI Indigo R4000 workstation measured for the direct decomposition of a three-dimensional aero-dynamic blade with 7552 elements. This figure points out that the Greedy scheme is two orders of magnitude cheaper than RSB.
This fact has been observed for every mesh tested. Decompositions with other algorithms are performed with a CPU time ranging between the Greedy and the RSB CPU times.

The quality of decompositions and the low compute cost makes the Greedy algorithm the best-suited for the two-step approach. However, we must be very careful with such general results. In practice, the experience of the user will guide the choice of an algorithm.

4.2. Optimization

The initial decompositions have been optimized with the three heuristics. We have performed each optimization 10 times, and the best decomposition has been retained. However, the optimizers have proven to be very robust and, in practice, two or three runs are sufficient to improve the initial decomposition significantly. Moreover, the tests have proven that differences in quality of the optimizers are not significant. The penalty parameter $\alpha$ is set to 0.9 and decompositions with identical number of elements per subdomains have been considered.

The average percentage of interface node reduction for the entire set of tests (every grid and every optimizer) is shown in Figure 2. The reduction rises up to 20% for 64 subdomains. It seems to increase with the number of elements or with the number of subdomains.

Figure 3 shows the computation time for one run of each optimizer. The added CPU cost due to the optimization remains affordable compared to direct decomposition costs shown in Figure 1.

Sample performance results are given in Table 2 for a three-dimensional aerodynamic blade. The RSB and Greedy algorithms are used to provide initial decompositions and the SA heuristic is used to improve them. The communication timings are reported for one time-step of an Euler flow computation on the iPSC-860.

The above results highlight the potential of the proposed two-step procedure. The SA
algorithm is shown to reduce the interface size produced by the RSB scheme by over 30\%, and communication costs by over 20\%. The use of the Greedy scheme as initial algorithm, followed by SA, tends to outperform the use of RSB without any optimizer. The average reduction between RSB and G-SA equals 20\% for the number of interfaces nodes, and 15\% for the communication costs. Finally, we must remember that the CPU requirements of RSB and G-SA are of the same order.

<table>
<thead>
<tr>
<th>$N_s$</th>
<th>number of interface nodes</th>
<th>communication costs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RSB</td>
<td>RSB-SA</td>
</tr>
<tr>
<td>16</td>
<td>2169</td>
<td>1657</td>
</tr>
<tr>
<td>32</td>
<td>3031</td>
<td>2296</td>
</tr>
<tr>
<td>64</td>
<td>3982</td>
<td>3062</td>
</tr>
<tr>
<td>128</td>
<td>5203</td>
<td>4028</td>
</tr>
</tbody>
</table>

Table 2. Sample performance results for a 3D blade. Reduction of the interface size and related reduction of the communication costs for an Euler flow problem.

5. CONCLUSION

We have developed a two-step approach for the automatic domain decomposition of non-structured grids. This approach consists of a direct method followed by a non-deterministic optimization.

The optimization step provides significantly improved partitions at a reasonable cost. Furthermore, the heuristics are able to take into account constraints and complex objectives in a rather natural manner. As a side effect, the optimization step produces
decompositions with geometrically smooth interfaces, which is crucial to some modern numerical algorithms based on domain decomposition.

The above algorithms have been implemented in the TOP/DOMDEC software developed at the University of Colorado at Boulder. This software is an interactive package for mesh partitioning and parallel processing. It offers several state-of-the-art graph decomposition algorithms in a user friendly environment.

REFERENCES

9. Y. G. Saab and V. B. Rao, “Combinatorial Optimization by Stochastic Evolu-