Transactions on Information and Communications Technologies vol 8, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Partitioning unstructured meshes using a distributed optimization model

N. Bouhmala, K. Ghedira, H.H. Naegeli
Institute of Computer Science and Artificial Intelligence, Emile-Argand 11, 2007 Neuchâtel, Switzerland

Abstract

Large meshes computations arise in many large-scale scientific and engineering problems, including finite volume methods for computational fluid dynamics, and finite element methods for structure analysis. If these meshes have to be solved efficiently on distributed memory parallel processors, a partitioning strategy should be designed so that on the one hand, processors have approximately equal work to do, and on the other hand inter-processor communication is minimized. In this paper we introduce a distributed optimization model combining Multi-agent systems and simulated annealing for the mesh partitioning problem.

1 Introduction

Many large-scale computational problems are based on unstructured computations domains. Among such problems, unstructured grid calculations based on finite volume methods in computational fluid dynamics (CFD), and structural analysis problems based on finite element approximations. CFD is potentially an extremely powerfull tool for flow applications in, for example, the aeronautical, aerospace, automotive and chemical industries. The majority of CFD methods are based on the resolution of a set of partial differential equations such as the Euler equations for inviscid flow, or the Navier-Stokes equations for viscous flow, that describe the continum behavior of the fluid [7]. The numerical simulation of complex CFD applications requires a vaste amount of computer power.
One of the major problems with implementing such problems on a distributed memory machine is how to partition these meshes into several submeshes, which are then mapped to distinct processors (mapping problem). In the sequel a submesh will be referred to as a partition.

This paper is organized as follows: section 2 introduces the mesh partitioning problem and name some well known methods suggested which attempt to find an approximation to the best partitioning; section 3 introduces the model and finally section 4 gives some possible future work.

2 The mesh partitioning problem

Structured and unstructured meshes are represented as undirected graphs G using sparse matrix data structure. The numerical algorithms that operate on these meshes involve the repeated application of the same computation at the variables located at the vertices of G, with data dependencies between the variables given by the edges of G. Parallelization of this type of problems is achieved by using the data parallelism paradigm: the data structure of the problem is distributed over the processors of the parallel machine and each processor is responsible for the work on its data. The distribution of data must be done in such a way that the total execution time is minimized. Therefore, the partitioning scheme should be designed with the following criteria:

- Load balance the problem: This is achieved by assigning an equal amount of computational work to each processor. If the same amount of work is associated with all the elements, equal distribution of elements ensures a good load balance.
- Minimize the quantity of communication : This means minimizing the length of the boundary of each processor's subregion.

The execution time spent in the partitioning scheme phase, should be reasonably small with respect to the total execution time. Several approaches have been proposed for the mesh partitioning problem which is known to be NP-hard, for instance: recursive coordinate bisection and recursive spectral bisection [14], a greedy algorithm [5], integer linear programming [1], and stochastic techniques such as simulated annealing [11], and genetic algorithms [12].

Moreover if the mapping problem [6] is taken into account, the distance that information must be communicated between processors has to be minimized: Any two submeshes which share a common boundary should be, as much as possible, assigned to nearest neighbor processors. Otherwise communication has to go through multiple processors, resulting in a degradation of the system performance because of high overhead cost.

3 Distributed Optimization Model

3.1 Why distributed optimization ?

Many researchers have been using centralized approaches to solve many combinatorial optimization problems, all the more reason to explore another direction : multi-agent systems. Indeed the multi-agent approach opens a new way to solve diverse problems in terms of cooperation, conflict and concurrence within a society of agents. Each agent is an autonomous entity which is asychronously able to acquire and comunicate information from or to its environment. On the basis of this inormation an agent can reason and consequently undertake decisions. The model we propose combines simulated annealing and multi-agents systems : each agent tries to optimize its local cost function by using its own simulated annealing that it locally controls. We advocate for this approach, since it has been successfully applied to Constraint Satisfaction Problems [9] [10], to the Resource Allocation Problem (RAP) [8] and to the flow shop scheduling problem [4].

Since the mesh partitioning problem viewed as a graph partitioning problem, can be formulated as a RAP or a constrained optimization problem , the choice for this approach seems to be a good one. In this connection, a distributed

graph partitioning problem has been defined in [3] as an extension of the classical graph partitioning problem.

3.2 The Model

Generally speaking, the model we propose for the mesh partitioning problem, consists of interacting agents. In the first place each agent is assigned a partition to work with, and attempts on the one hand to minimize the number of its external edges (edges that connect an agent with other agents), and on the other hand to get the average workload, by communicating with its partner which will be determined by the partnership mechanism.
Moreover, a mechanism, called locking mechanism, allows all the pairs of partners generated by the partnership mechanism, to work independently. This work is performed as follows: The two partner agents that are supposed to work together select the vertices that will possibly be exchanged.
Thereafter, each agent performs its own simulated annealing and takes a temporary decision. A conflict decision may occur between partners, the reason why we introduce the OR logic based-decision mechanism.

The algorithm for the proposed model is shown in figure 1. The underlying agents, their behavior and the different mechanisms will be detailed in the following subsections.

```
Algorithm:
begin
        Initial partitioning;
        repeat
                Partnership mechanism;
                For each agent (simultaneously);
                        Locking mechanism;
                        generate a new state (migrate random vertices);
                        evaluate variation cost;
                        apply accept/reject test (temporary decision);
                        exchange temporary decisions;
                        OR logic based-decision mechanism;
        until stop;
end.
```

Figure 1: The algorithm for the proposed model.

3.2.1 Partnership mechanism Suppose that our society of agents have an interconnection pattern similar to a grid (see figure 2). We suppose that initially the graph has been partitioned and the partitions have been assigned to different agents according to some heuristic. The strategy goes by forming a partnership between agents. This partnership has to guarantee the non-interference property : Interference occurs when an agent promises the same vertex to two or more other different agents.

To ensure the non-interference property between agents, one has to generate a set of boundaries no two of which share a common agent. Figure 2a and figure 2b show 8 boundaries generated by he partnership mechanism (PM) with the property of non-interference. After applying the PM, the acquaintances of each agent are limited to one partner. Moreover the PM is dynamic, in the sense that the set of boundaries that are generated each time, may be different. Adopting this strategy, all boundaries are subject to changes (see figure 3).

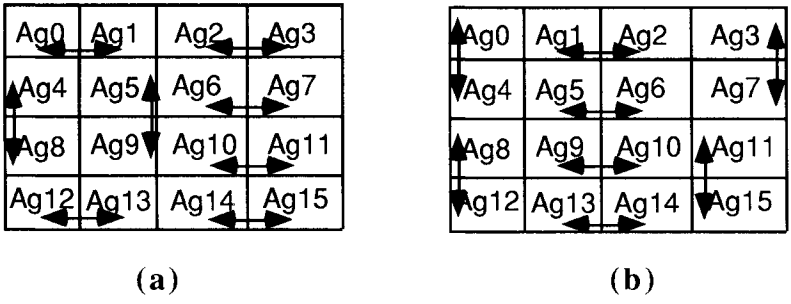

Figure 2: Boundaries with the non-interference property

```
partnership mechanism algorithm;
/* Input : A List of N agents, ListAgents. */
/*For each agent, a list of its acquaintances.*/
/* Output :N/2 pairs of partners*/
begin
      while ListAgent ≠ Ø do
      {
      Agselect ----> random agent from ListAgent;
      Agsadj ----> random agent from acquaintancelist (Agselect);
      record this choice ; /* pair of partners */
      identify the agents acquainted to Agselect and Agsadj;
      remove Agselect and Agsadj from acquaintancelist for each
      identified agent;
      remove Agsadj and Agselect from ListAgent;
      }
end;
```

Figure 3: Partnership mechanism algorithm.

3.2.2 Structure of an agent Each agent is defined by :
• Its partition (subset of vertices).
• Its acquaintances that are defined by its nearest neighbor agents: A nearest neighbor of an agent is defined as the agent sharing a boundary with it. In the interconnection pattern we have proposed (see figure 2), each agent has at most 4 nearest neighbors. Each time the PM is performed, only one of them is chosen as a partner.
• Its local cost function: The search for optimality must be guided by a cost function that reflects the vital parameters to be optimized. Since the total resources consumed by an agent consists of two components (a local load and a communication cost), our cost function is formulated as follows:

$$Cost(Ag_k) = \left(W_k - \overline{W}\right) + \mu \sum_{l=1, l\neq k}^{N_k} \left(\sum_{i\in Ag_k, j\in Ag_l} c_{ij} \right) \qquad (1)$$

Where :

- Cost (Ag_k) : the cost associated with $agent_k$.
- W_k: the work associated with $agent_k$..
- $W_k = \sum_{i \in Ag_k} w_i$, where w_i : the computation time associated with vertex i.
- $\overline{W}$: the average workload.
- μ : imbalance factor.
- c_{ij}: the weight of the edge joining vertices i and j.
- N_k : number of agent communicating with $agent_k$.

The first term of the cost function serves to minimize the imbalance of load, while the second term attempts to minimize the communication cost for the $agent_k$., while keeping the correct proportion between both terms by accounting for the factor μ

• Its own optimization tool (simulated annealing): Each agent tries to optimize its own local cost function eqn (1). The local implementation of this tool will be further detailed.

3.2.1 Locking mechanism In order to ensure the consistency of the local optimization done at the level of each agent, a locking mechanism is introduced.

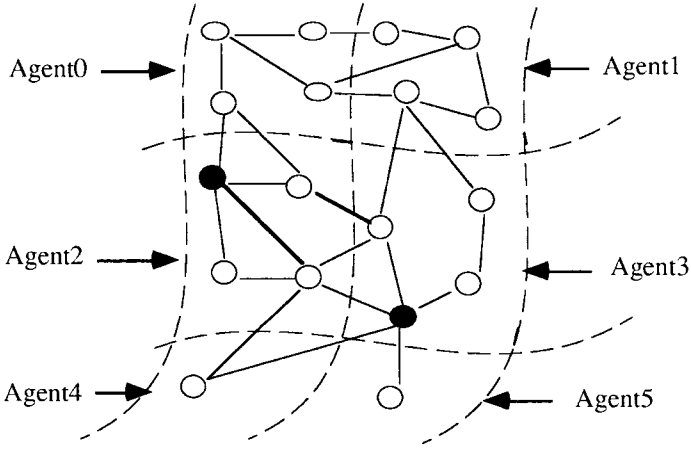

Figure 4: The problem associated with the exchange of vertices.

To guarantee the independence of all the pairs of agents, the vertices that should be selected to be exchanged within each pair, are the ones that don't affect the cost of other pairs.

Suppose a given graph (see figure 4) has been assigned initially to 6 agents organized in three pairs $agent_0$, $agent_1$, $agent_2$, $agent_3$, $agent_4$ and $agent_5$. If the vertices colored in black are to be exchanged between $agent_2$ and $agent_3$, this exchange will affect $agent_0$, $agent_4$, $agent_5$ respectively.

Before these agents can evaluate their local cost function and decide whether to accept or to reject for instance an exchange according to their simulated annealing mechanisms, additional communication has to go between: ($agent_0$, $agent_2$), ($agent_3$, $agent_4$), ($agent_3$, $agent_5$). Therefore we adopt the strategy to lock these "noisy vertices" black vertices in order to guarantee the consistency.

3.2.3 Local simulated annealing Simulated annealing (SA) is a stochastic general-purpose combinatorial optimization technique [13], which stochastically simulates the slow cooling of a phisical system. The idea is that there is a cost function which associates a cost with a state of the system, a temperature, and various ways to change the state of the system. The basic feature of the SA is the ability to explore the search space of the problem allowing controlled hill-climbing moves (moves that increase the cost of the solution) in an attempt to reduce the probability of becoming stuck at a local minimum (maximum).

The SA implemented in our case proceeds as follows: Given an agent, let $S_{current}$ and $C_{current}$ denote respectively the current configuration and its associated cost. A new partition is generated by a perturbation of $S_{current}$, resulting in a new local cost, C_{new}. Each agent generates a new configuration either, by an exchange of vertices with one of its acquaintances agents, or by a displacement of a vertex from (resp. to) one of its acquaintances if it is underloaded (resp. overloaded). The change in cost $\Delta C = C_{new} - C_{current}$ is evaluated by each agent. If ($\Delta C <= 0$) the move is unconditionally accepted; if ($\Delta C > 0$) the move is accepted with a probability $\exp(-\Delta C/T)$, where T denotes the agent local temperature. To avoid a decision conflict between two agents, sharing a common boundary, a coordination mechanism is added and will be dealt with in the following section

3.2.4 OR logic based-decision Since each agent is running its own SA trying to satisfy its own local cost function, a decision-problem arises when the two agents which are partners have two different decisions. To deal with this problem, we propose the following simple procedure : Two agents agree upon accepting the proposed change if either of them has an accept decision. In case of two different decisions, the agent with an accept decision will be labelled as master , while its partner will be labelled as slave. The agent labelled master will impose in fact the new boundary to its partner.

One might ask the following question : Why choose the OR logic based-decision ? The idea behind choosing this strategy is to let the system to explore many configurations in the search space. If only one agent is in favour of a change, this change will take place. If the "AND" logic were adopted for instance, two partners will not modify their current partitions unless both agents are in favour of it, making the search by SA inefficient.

4 Conclusion

In this paper, a multi-agent model combined with simulated annealing for the mesh partitioning problem is introduced for the first time. Each agent is responsible for its partition/submesh (subset of vertices) and tries to minimize its external edges and load balance its partition by using its own simulated annealing.

Thereafter, this society of agents has been decomposed, thanks to a set of mechanisms (partnership and locking), to a set of pairs of agents (called partners) totally independent from local optimization point of view.

A third mechanism based on the OR logic has been added at the level of each agent to avoid conflict decision within each pair.

At the present time, the implementation of the approach is under progress. The experimentation includes testing the proposed approach on a wide class of graphs, and on meshes arising from real life applications.

Acknowledgement

This project is financed by the Swiss National Funds of the Scientific Reseach under contract # 21-40757.94.

References

1. Barahona, F. & Casari, A. On the magnetisation of the ground states in two dimensional ising spin glasses, *Comp. Phys. Communications,*, 49:417, 1988.
2. Bomholt, L. & Leyland, P. Implementation of Unstructured finite Element Codes on Different Parallel Computers, *Proceedings, Parallel CFD '93*, Paris 1993.
3. Bouhmala, N., Ghedira, K. & Naegeli, H.H. How to partition a graph by a Multi-agent approach based on a hybrid optimization tool, *Proceedings of the Fifth Scandinavian Conference on AI*, Trondheim, Norway 1995.
4. Daouas, T., Ghedira, K. & Muller, J-P. Distributed flow shop scheduling problem: Global versus local optimization, *Proceedings of the first International Conference on Multi-Agent Systems*, San Francisco 1995.
5. Farhat, C . A simple and efficient automatic FEM domain decomposer, *Computers and Structures,*, 28, 579-602, 1989.
6 Farhat, C. On the mapping of massively parallel processors onto finite element graphs, *Computers and Structures* , 1989, Vol. 32, No.2, 347-353 .
7. Freskos, G. & Penanhoat, O. Numerical Simulation of the Flow Field Around Supersonic Air-intakes, *ASME Paper* 92-GT-206 1992.
8. Ghedira, K. & Verfaillie, G. A multi-Agent model for the resource allocation problem: a reactive approach, *Proceedings of European Conference of Artificial Intelligence*, Austria 1992.
9. Ghedira, K. A distributed Approach to Constraint Satisfaction Problems, *Proceedings of European workshop of Modelling Autonomous Agents in Multi-Agent World,* Denmark 1994.
10. Ghedira, K. Distributed Simulated Re-annealing for Dynamic Constraint Satisfaction Problems, *Proceedings of International Conference on Tools with Artificial Intelligence*, USA 1994.
11. Jhonson, D.C., Aragon, C.R., McGeeoh & L.A., Schevon, C. Optimization by Simulated Annealing: An Experimental Evaluation, part I: (The GPP), *Operations Research* , 37:865, 1989.
12. Höhm, C. Heuristic Neighbourhood Search Operators For Graph Partitioning Tasks, *Proceedings of the tenth International Conference on Systems Engineering,* 469-476, 1994.
13. Kirkpatrick, S., Gelatt, C. & Vecchi, M. Optimization By Simulated Annealing, *Science*, 1983, Vol.220, No.4598, 671-680.
14. Simon, H.D. Partitioning of unstructured problems for parallel processing, *Computing Systems in Engineering*, 1991, Vol.2, No. 2/3, 135-148,1991.