Parallel FDTD Computations Based on Macro Data Flow Graphs Transformed by Genetic Algorithms

Adam Smyk[1], Marek Tudruj[12],

1. Polish-Japanese Institute of Information Technology,
86 Koszykowa Str., 02-008 Warsaw, Poland
2. Institute of Computer Science, Polish Academy of Sciences,
21 Ordona Str., 01-237 Warsaw, Poland,
{asmyk,tudruj}@pjwstk.edu.pl

1 Extended abstract

The Finite Difference Time Domain (FDTD) method belongs to numerical applications that can be described by irregular data patterns or use irregular data structures. This method is based on an iterative algorithm, which is used to simulate high frequency electromagnetic wave propagation. Such simulation is performed by solving Maxwell equations transformed into their differential form. In order to perform such a simulation in a multiprocessor system, the whole computational area must be divided into subareas assigned to particular processors. In order to balance computational load, size of all computational subareas must be equalized. To minimize communication volume, length of the border between each two adjacent subareas must be minimal.

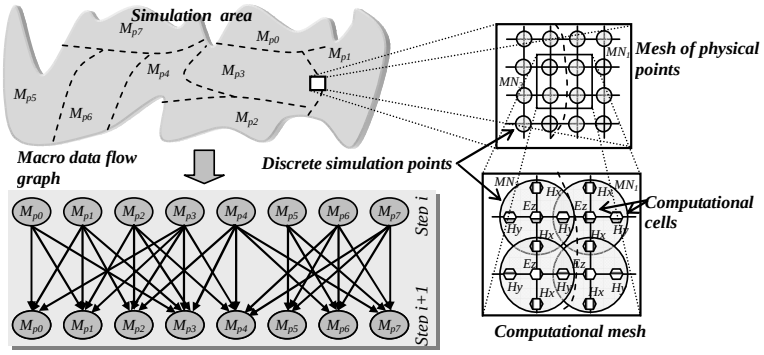

Fig. 1. Example of the MDFG for irregular computational area

In this paper, we present and compare two algorithms, which optimize FDTD [3] computations performed in multiprocessor systems for irregular wave propagation areas. We have assumed that optimized computation and communication patterns are given as a macro data flow graph (MDFG) generated by a transformation of an initial data flow graph of computations, see Fig.1. The compared

algorithms are based on genetic approach with standard genetic operators (selection, crossover, mutation) but with special meaning of chromosomes and different optimization methods, see Fig.2.

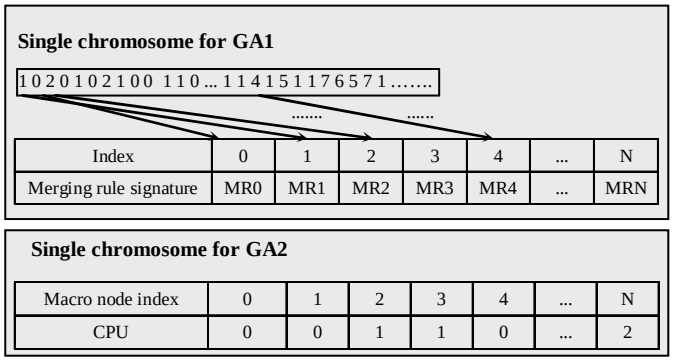

Fig. 2. Meaning of chromosomes for two tested genetic algorithms

In the first algorithm (GA1), a chromosome represents a complete data flow node merging algorithm given as a sequence of merging operations. Execution of each operation reduces the number of nodes by one, so after execution of the whole chromosome, the number of macro nodes will be equal to the number of processors. We have tested several kinds of merging methods. The chosen merging methods are presented in Table 1 and short descriptions for all tests performed for GA1 are presented in Table 2.

Table 1. Chosen merging rules for GA1

Id	Rule priority	Description
MR0	Computational load balancing	The two least loaded adjacent nodes will be merged
MR1	Computational load balancing	The most loaded node will be merged with the least loaded adjacent node
MR2	Computational load balancing	The least loaded node will be merged with the most loaded adjacent node
...	...	...
MR8	Computational load balancing with edge cut reduction	The least loaded node will be merged with the adjacent node with the biggest communication volume
MR9	Computational load balancing with edge cut reduction	The least loaded node will be merged with the adjacent node with the lowest communication volume

In the second algorithm (GA2), a chromosome corresponds to an assignment of macro nodes to processors, see Fig.2. In this algorithm, we have introduced a mixed fitness function composed of two contradictory sub-functions used simultaneously: the cut-min value function CMFF (it represents the number of edges connecting two macro nodes assigned to two different processors [1]) and

the difference between the maximally and minimally loaded macro nodes in an individual, MAXMINFF. The first value determines the total volume of communication while the second one determines the maximal computational load imbalance. The GA1 method based only on the MAXMINFF. In the GA2 mainly the MAXMINFF has been used, but every k iterations CMFF has been applied, see Fig.5.

Table 3. Tests descriptions for GA1

Test symbol	Description
Test1	Test performed for the following (manually chosen) merging rules: MR0,MR1,MR2,MR3
Test2	Test performed for all merging rules
Test3	Test performed for all merging rules with random re-selection of the best 5% of individuals
Test4	Test performed for all merging rules with random re-selection of the best 5% of individuals and additionally another 5% of individuals are replaced by the best individual
Test5	Test performed for all merging rules with random re-selection of the best 1% of individuals and additionally another 1% of individuals are replaced by the best individual
Test6	Test performed for all merging rules with re-selection: 5% of individuals are replaced by the best individual in population
Test7	Test performed for merging rules: MR0,MR1,MR4,MR7,MR8 (manually chosen); with re-selection: 5% of individuals are replaced by the best individual in population
Test8	Test performed for merging rules: MR0,MR1 (manually chosen); with re-selection: 5% of individuals are replaced by the best individual in population

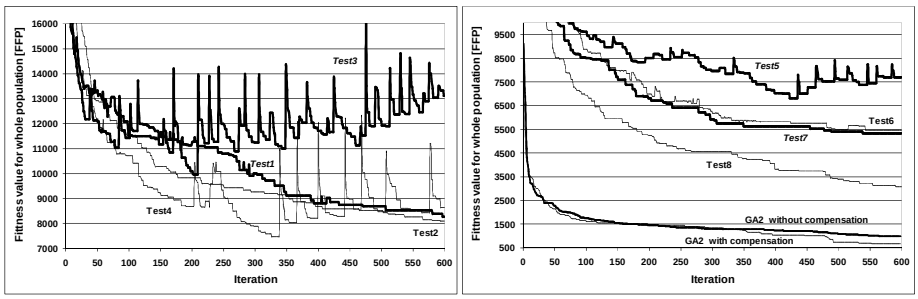

Fig. 3. Convergence results for GA1 and GA2 algorithms

We have tested both algorithms by simulation for multiprocessor systems with shared and distributed memory with MPI and RDMA rotating buffers communication [2]. The experimental results have shown, which algorithm allows faster adjustment of its behavior to the requirements of the computational system. We performed several experiments to test our genetic algorithm. The

description of each test is presented in Table 3. All experiments were done for the constant size of population (50 individuals) and for constant number of iterations (600). Values of the fitness function for all individuals are added up to give a fitness value for the whole tested population (FFP). It is used as an indicator of the convergence of the genetic algorithms. Our experiments have shown, that the GA2 with the mixed fitness function produced better FDTD graph partitioning in comparison to GA1. Moreover, it allows dynamically adjust a mixed fitness function according to the requirements of the computational system.

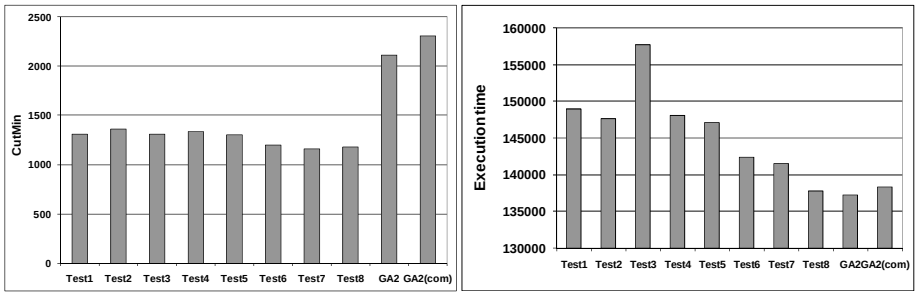

Fig. 4. Cut min value and FDTD method execution time for GA1 and GA2 algorithms

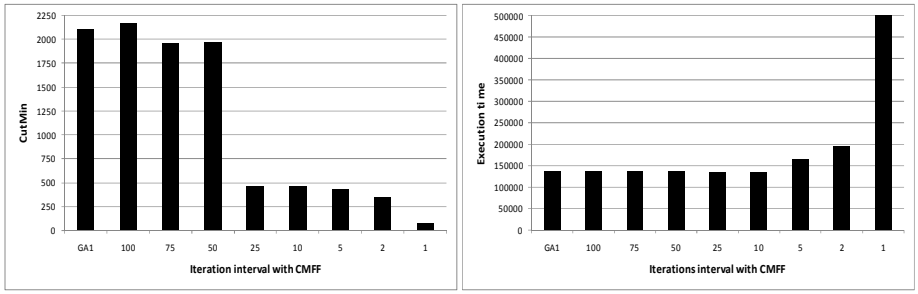

Fig. 5. Mixed fitness function influence on the cut min value and the FDTD method execution time (GA2)

References

[1] M.S. Khan, K.F. Li, "Fast Graph Partitioning Algorithms", Proceedings of IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Victoria, B.C., Canada, May 1995, pp. 337-342

[2] A.Smyk, M.Tudruj, "RDMA Control Support for Fine-Grain Parallel Computations", PDP 2004, La Coruna, Spain

[3] A.Smyk, M.Tudruj, "Parallel Implementation of FDTD Computations Based on Macro Data Flow Paradigm", PARELEC 2004, September 7-10, Dresden, Germany