Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 1993–2001
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Graph grammar-based multi-thread multi-frontal parallel solver with trace theory-based scheduler

Paweł Obrok, Paweł Pierzchała, Arkadiusz Szymczak, Maciej Paszyński

Department of Computer Science
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Cracow, Poland

Abstract

The paper presents the graph grammar based multi-thread multi-frontal parallel direct solver for one and two dimensional Finite Difference Method (FDM). The multi-frontal solver algorithm has been expressed by graph grammar productions. Each production represents an atomic task that internally must be executed in serial. The sequence of graph grammar productions modeling the execution of the solver has been associated with the alphabet for the trace theory analysis. The dependency relation between tasks has been introduced based on the analysis of the solver algorithm. The sequence of productions has been transformed into the Foata Normal Form (FNF). The parallel solver algorithm has been implemented and tested on NVIDIA Cuda multi-core graphic card. The tasks have been scheduled according to the classes in the FNF.

Keywords: multi-frontal direct solver, theory of concurrency, graph grammar, trace theory, scheduling, finite difference method

1. Introduction

The multi-frontal solver algorithm is the most sophisticated version of the direct solver [5]. The multi-frontal solver constructs an elimination tree by using the nested dissection algorithm [6], and solves the problem recursively traveling the nodes of the tree. In this paper the multi-frontal solver algorithm has been expressed by Composite Programmable graph grammar (CP-graph grammar). The CP-graph grammar originally designed for modeling composite structure [2, 3, 4] has been recently used for modeling adaptive Finite Element Method [7, 8]. In this paper we focus on the expression of the multi-frontal solver algorithm dedicated for Finite Difference Method (FDM) computations by means of the CP-graph grammar. The atomic tasks, that must be executed in serial, such as construction of a single frontal matrix or elimination of a single row from the matrix, are expressed by graph grammar productions.

The execution of the solver algorithm can be modeled then as a sequence of graph grammar productions. A single graph grammar production represents a single undividable task, that must be executed in sequential. Thus, the expressing of the solver by the graph grammar exploits the concurrency of the solver algorithm. The trace theory [1]

Email address: paszynsk@agh.edu.pl (Maciej Paszyński)
URL: http://home.agh.edu.pl/~paszynsk (Maciej Paszyński)

1877-0509 © 2012 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.04.223

can be applied for constructing the task scheduler for the shared memory architecture parallel machine. The sequence of graph grammar productions modeling the solver algorithm is represented as the alphabet for the trace theory. The inter-tasks dependency relation is obtained by analysing the solver algorithm. Having the alphabet and the dependency relation, the sequence of graph grammar productions is transformed into the Foata Normal Form (FNF) [1]. The FNF is finally used by the scheduler. The tasks are executed in groups of independent tasks, associated with classes from FNF. The presented solver has been implemented and tested on NVIDIA Cuda multi-core graphic card.

2. Multi-frontal solver expressed by CP-graph grammar

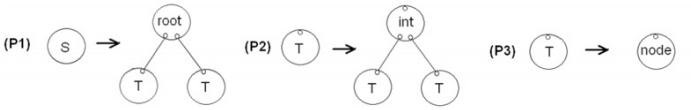

Figure 1: Set of graph grammar productions **P1**, **P2**, **P3** for the generations of one dimensional computational grid.

The operations performed by the solver have been expressed by CP-graph grammar.

The first set of graph grammar productions, presented in Figure 1, is responsible for the generation of the computational grid. The computational grid is represented by a CP-graph. The generation of the computational grid is expressed as the sequence of graph grammar productions. The generation starts from a single vertex initial graph, denoted by **S** symbol. Each graph grammar production consists of left-hand-side and right-hand-side graphs. The left-hand-side graph is localized on the current CP-graph graph and replaced by the right-hand-side. Thus, some graph grammar productions can be executed over several parts of the CP-graph. The number of executed graph grammar productions is denoted by the power in the sequence representing the application of graph grammar productions. The exemplary 1D, obtained by execution of the sequence of graph grammar productions **(P1)**-**(P2)**^{2}-**(P2)**^{4}-**(P3)**^{8} is presented in Figure 2. Let us trace the solver execution for the simple benchmark problem: Find the temperature

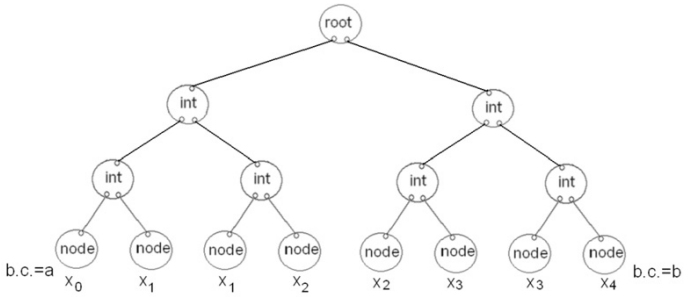

Figure 2: 1D grid obtained by executing productions **(P1)**-**(P2)**^{2}-**(P2)**^{4}-**(P3)**^{8}

distribution $R \ni x \mapsto u(x) \in R$ such that

$$\frac{d^2u}{dx^2} = 0 \text{ for } x \in [0,1] \qquad u(0) = 0 \qquad u(1) = 20. \tag{1}$$

The problem can be simply discretized by using finite difference method as

$$u_0 = 0 \tag{2}$$

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = 0 \text{ for } i = 1, ..., N-1 \tag{3}$$

$$u_N = 20 \tag{4}$$

where $h = \frac{1}{N}$ and $u_i = u(x_i) = u\left(\frac{i}{N}\right)$. The domain can be partitioned into sub-intervals $[0,1] = \bigcup_{i=1}^{N}[x_{i-1}, x_i]$. The equation (3) is related with i-th node of the grid, and must be partitioned now into two equations, associated with sub-intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$. Thus, a single internal sub-interval $[x_i, x_{i+1}]$ has the following system of two equations associated

$$\frac{u^I_{i-1} - u^I_i}{h^2} = 0, \qquad \frac{u^{II}_{i+1} - u^{II}_i}{h^2} = 0, \tag{5}$$

while the first sub-interval $[x_0, x_1]$

$$u^I_0 = 0, \qquad \frac{u^{II}_1 - u^{II}_0}{h^2} = 0, \tag{6}$$

and the last sub-interval $[x_{N-1}, x_N]$

$$u^I_N = 20, \qquad \frac{u^{II}_N - u^{II}_{N-1}}{h^2} = 0. \tag{7}$$

where $u_i = u^I_i + u^{II}_i$. Thus, the solver algorithm can be expressed in the following steps. First, the local matrices associated with particular sub-intervals are created. For the first sub-interval $[x_0, x_1]$ the system

$$\begin{bmatrix} 1 & 0 \\ \frac{1}{h^2} & -\frac{1}{h^2} \end{bmatrix} \begin{bmatrix} u_0 \\ u^I_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8}$$

is created, for the last sub-interval $[x_{N-1}, x_N]$ the system

$$\begin{bmatrix} -\frac{1}{h^2} & \frac{1}{h^2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u^{II}_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} 0 \\ 20 \end{bmatrix} \tag{9}$$

is created, and for any internal sub-interval $[x_i, x_{i+1}]$ the system

$$\begin{bmatrix} -\frac{1}{h^2} & \frac{1}{h^2} \\ \frac{1}{h^2} & -\frac{1}{h^2} \end{bmatrix} \begin{bmatrix} u^{II}_i \\ u^I_{i+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{10}$$

is created. This is expressed by graph grammar production **(A1)**, **(AN)** and **(A)** presented in Figures 3 and 4. These

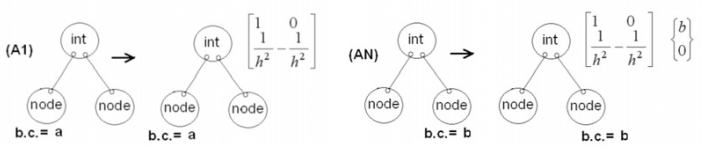

Figure 3: Graph grammar productions **(A1)** and **(AN)** constructing local matrices for the first and the last intervals, involving Dirichlet boundary conditions.

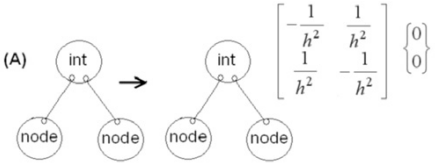

Figure 4: Graph grammar production **(A)** constructing local matrix for an internal interval.

graph grammar productions actually only attribute graph vertices by linear systems of equations. Notice, that all these graph grammar productions can be executed in parallel. The next step of the solver algorithm, is to travel recursively the elimination tree presented in Figure 2, merge two linear systems of equations from son nodes into one new system constructed at the father node, and eliminate one fully assembled equation. This is expressed by graph grammar

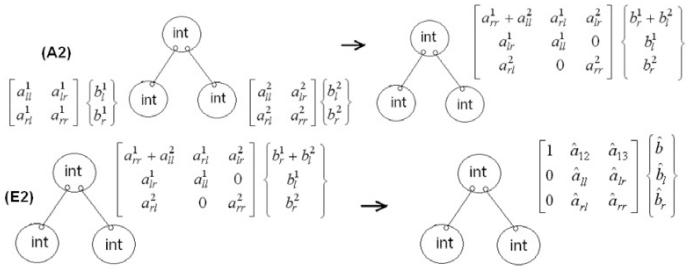

Figure 5: Graph grammar productions **(A2)** and **(E2)** merging two matrices related to son nodes, and eliminating fully assembled equation representing the common node.

productions **(A2)** and **(E2)** presented in Figure 5. E.g. merging of two matrices from leaves associated with internal sub-intervals, can be expressed as

$$\left[\begin{array}{cc} -\frac{1}{h^2} & \frac{1}{h^2} \\ \frac{1}{h^2} & -\frac{1}{h^2} \end{array}\right]\left[\begin{array}{c} u_{i-1}^{II} \\ u_i^{I} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \end{array}\right] \bigoplus \left[\begin{array}{cc} -\frac{1}{h^2} & \frac{1}{h^2} \\ \frac{1}{h^2} & -\frac{1}{h^2} \end{array}\right]\left[\begin{array}{c} u_i^{II} \\ u_{i+1}^{I} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \end{array}\right] =$$
$$\left[\begin{array}{ccc} -\frac{2}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} \\ \frac{1}{h^2} & -\frac{1}{h^2} & 0 \\ \frac{1}{h^2} & 0 & -\frac{1}{h^2} \end{array}\right]\left[\begin{array}{c} u_i \\ u_{i-1} \\ u_{i+1} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right] \tag{11}$$

where $\bigoplus$ symbolizes the merging process. This corresponds to a single execution of **(A2)** production. Notice that the fully assembled equation associated with u_i variable is put as the first row of the resulting matrix. This is followed by a single execution of **(E2)** production eliminating the fully assembled row. The resulting 2×2 sub-matrix will be utilized for the merging process at father node.

$$\left[\begin{array}{ccc} -\frac{2}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} \\ \frac{1}{h^2} & -\frac{1}{h^2} & 0 \\ \frac{1}{h^2} & 0 & -\frac{1}{h^2} \end{array}\right]\left[\begin{array}{c} u_i \\ u_{i-1} \\ u_{i+1} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right] \mapsto \left[\begin{array}{ccc} -\frac{2}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} \\ 0 & \frac{1}{h^2} & \frac{2}{h^2} \\ 0 & \frac{2}{h^2} & \frac{1}{h^2} \end{array}\right]\left[\begin{array}{c} u_i \\ u_{i-1} \\ u_{i+1} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right] \tag{12}$$

Finally, the root of the elimination tree is reached, the system of equations is merged, and solved, since all three equations are fully assembled at the root node. This is expressed by graph grammar productions **(Aroot)** and **(Eroot)** presented in Figure 6. The process of partial forward eliminations is followed by recursive backward substitutions.

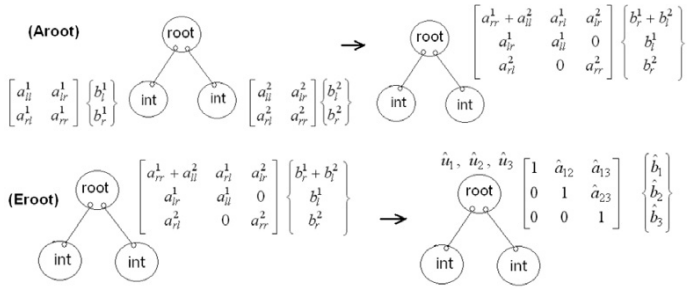

Figure 6: Graph grammar productions **(Aroot)** and **(Eroot)** merging two matrices related to son nodes of the root node, and performing full solution of the common interface problem related with the root node.

The solution obtained at the father node is used for partial backward substitutions executed at son nodes. This is expressed by graph grammar production **(BS)** presented in Figure 7. The solution obtained at father node is substituted on the right-hand-side, the identity 2×2 matrix is put on the left-hand side, and the backward substitution is executed

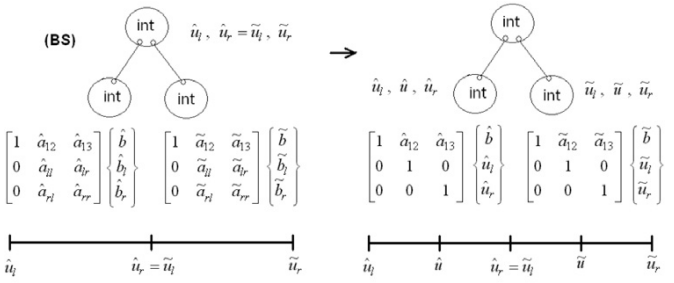

Figure 7: Graph grammar production **(BS)** executing partial backward substitutions within son nodes, by using partial solution retrieved from the father node.

at each son node. E.g. the backward substitution for the system (12) generated for tree leaves associated with internal nodes has the following form

$$\begin{bmatrix} -\frac{2}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} \\ 0 & \frac{1}{h^2} & \frac{2}{h^2} \\ 0 & \frac{2}{h^2} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} u_i \\ u_{i-1} \\ u_{i+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} -\frac{2}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ u_{i-1} \\ u_{i+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 20\frac{i-1}{N} \\ 20\frac{i+1}{N} \end{bmatrix} \tag{13}$$

The following section presents analysis of the exemplary problem solution according to the trace theory. The alphabet A of the traces consists of the following basic actions:

$$\begin{aligned} A = \{&(P1),\ (P2)^1,\ (P2)^2,\ (P2)^3,\ (P2)^4,\ (P2)^5,\ (P2)^6,\ (P3)^1,\ (P3)^2,\ (P3)^3,\ (P3)^4,\ (P3)^5,\\ &(P3)^6,\ (P3)^7,\ (P3)^8,\ (A1),\ (A)^1,\ (A)^2,\ (AN),\ (A2)^1,\ (A2)^2,\ (E2)^1,\ (E2)^2,\ (Aroot),\\ &(Eroot),\ (BS)^1,\ (BS)^2\}. \end{aligned}$$

The dependency relation D is defined as a transitive closure of the following set:

$$\begin{aligned} D = \{&\{(P1),\ (P2)^1\},\ \{(P1),\ (P2)^2\},\ \{(P2)^1,\ (P2)^3\},\ \{(P2)^1,\ (P2)^4\},\ \{(P2)^2,\ (P2)^5\},\\ &\{(P2)^2,\ (P2)^6\},\ \{(P2)^3,\ (P3)^1\},\ \{(P2)^3,\ (P3)^2\},\ \{(P2)^4,\ (P3)^3\},\ \{(P2)^4,\ (P3)^4\},\\ &\{(P2)^5,\ (P3)^5\},\ \{(P2)^5,\ (P3)^6\},\ \{(P2)^6,\ (P3)^7\},\ \{(P2)^6,\ (P3)^8\},\ \{(A1),\ (A2)^1\},\\ &\{(A)^1,\ (A2)^1\},\ \{(A)^2,\ (A2)^2\},\ \{(AN),\ (A2)^2\},\ \{(A2)^1,\ (E2)^1\},\ \{(A2)^2,\ (E2)^2\},\\ &\{(E2)^1,\ (Aroot)\},\ \{(E2)^2,\ (Aroot)\},\ \{(Aroot),\ (Eroot)\},\ \{(Eroot),\ (BS)^1\},\\ &\{(Eroot),\ (BS)^2\}\} \cup \{\text{each action is dependent on itself}\}. \end{aligned}$$

The following sequence of actions generates the elimination tree:
$(P1) - (P2)^1 - (P2)^2 - (P2)^3 - (P2)^4 - (P2)^5 - (P2)^6 -$
$(P3)^1 - (P3)^2 - (P3)^3 - (P3)^4 - (P3)^5 - (P3)^6 - (P3)^7 - (P3)^8$
For such a sequence of actions the Foata Normal Form looks as follows:
$[(P1)][(P2)^1\ (P2)^2\ (P2)^3\ (P2)^4\ (P2)^5\ (P2)^6]$
$[(P3)^1\ (P3)^2\ (P3)^3\ (P3)^4\ (P3)^5\ (P3)^6\ (P3)^7\ (P3)^8]$

The following sequence of actions realizes the solver algorithm, i.e. partial forward eliminations and backward substitutions:
$(A1) - (A)^1 - (A)^2 - (AN) - (A2)^1 - (A2)^2 - (E2)^1 - (E2)^2 -$
$(Aroot) - (Eroot) - (BS)^1 - (BS)^2$
The Foata Normal Form for such a sequence looks as follows:
$[(A1)\ (A)^1\ (A)^2\ (AN)][(A2)^1\ (A2)^2][(E2)^1\ (E2)^2]$

$[(Aroot)][(Eroot)][(BS)^1\ (BS)^2]$

All actions in the same Foata set may be executed concurrently. Any two actions from different Foata sets must be executed sequentially.

3. Two dimensional finite difference

In 2D the generation of elimination tree does not differ from the one for 1D. The only difference is that each leaf of the elimination tree has assigned one vertical slice of the 2D domain. This section describes the solver algorithm for 2D: partial forward elimination and backward substitution. Let us consider a 2D version of the 1D exemplary problem described in section 2 - find a temperature scalar field u:

$$(0,1)^2 \ni (x,y) \mapsto u(x,y) \in R \tag{14}$$

such that:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{15}$$

or:

$$\frac{\partial^2}{\partial x^2} u(x_i, y_j) + \frac{\partial^2}{\partial y^2} u(x_i, y_j) = 0 \text{ where } i, j = 0, ..., N. \tag{16}$$

Figure 8 shows domain decomposition into four elements and the corresponding sought u values. The PDE can be

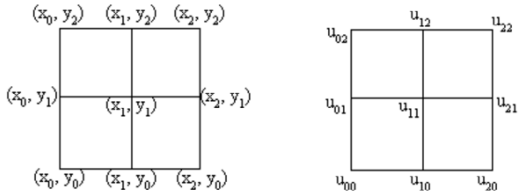

Figure 8: Domain decomposition into four elements and the corresponding sought u values.

approximated with the following difference equation:

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} = 0 \tag{17}$$

Continuing the example, two new child nodes are created under the root of the elimination tree. Each new child node represents a subsystem of equations corresponding to a vertical slice of the domain. This partitioning is presented in figure 9. In each child node the variables associated with the unique column are eliminated. Variables associated with the common column of the subdomains are propagated to the root node. The subsystems of equations from the child nodes are aggregated in the root node and variables associated with the common column are eliminated. This is illustrated in figure 10. Since this is the root of the elimination tree, a complete elimination is performed. Otherwise, a next common column would be propagated a level up.

The elimination process can be parallelized effectively. Let us put $M = 3N$ for the root of the elimination tree and $M = 2N$ for the child nodes in the elimination tree, where N is the mesh size and $N^2 = n$ is the total number of unknowns. Let us mark with $E_{R\ j-i}$ $(i, j = 1, \ldots, M)$ subtraction of i^{th} row (multiplied by appropriate factor) from j^{th} row in order to obtain zeros in j^{th} row below the main diagonal. Then all $E_{R\ j-1}$ $(j = 2, \ldots, M)$ can be executed in parallel, all $E_{R\ j-2}$ $(j = 3, \ldots, M)$ and so on. Speaking of graph grammar productions, they work analogously to the 1D case, i.e. they transform a graph node attribute being a matrix before the elimination step to the attribute being the same matrix after the elimination step.

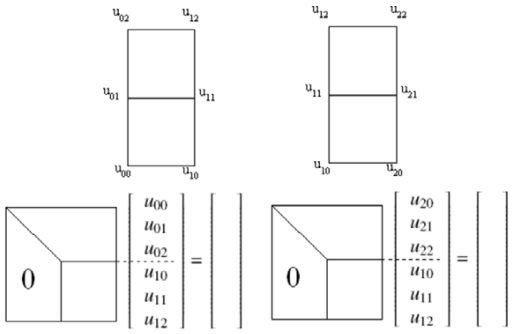

Figure 9: Partitioning the domain for the child nodes in the elimination tree.

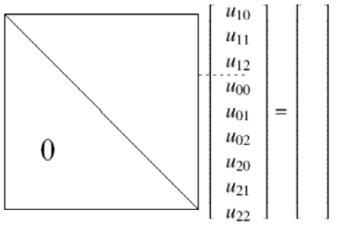

Figure 10: Aggregation of the complete system of equations in the root of the elimination tree.

Let us analyze the example of 2D partial forward elimination with the trace theory. The alphabet A consists of the following sets of actions:

$$\begin{aligned}
\text{E1} = \{&(E_{R\,2-1})^1,\ (E_{R\,3-1})^1,\ (E_{R\,4-1})^1,\ (E_{R\,5-1})^1,\ (E_{R\,6-1})^1,\ (E_{R\,3-2})^1,\ (E_{R\,4-2})^1,\ (E_{R\,5-2})^1,\\
&(E_{R\,6-2})^1,\ (E_{R\,4-3})^1,\ (E_{R\,5-3})^1,\ (E_{R\,6-3})^1\} \text{ - for the first child submatrix;}\\
\text{E2} = \{&(E_{R\,2-1})^2,\ (E_{R\,3-1})^2,\ (E_{R\,4-1})^2,\ (E_{R\,5-1})^2,\ (E_{R\,6-1})^2,\ (E_{R\,3-2})^2,\ (E_{R\,4-2})^2,\ (E_{R\,5-2})^2,\\
&(E_{R\,6-2})^2,\ (E_{R\,4-3})^2,\ (E_{R\,5-3})^2,\ (E_{R\,6-3})^2\} \text{ - for the second child submatrix;}\\
\text{E3} = \{&(A),\ (E_{R\,2-1})^3,\ (E_{R\,3-1})^3,\ (E_{R\,4-1})^3,\ (E_{R\,5-1})^3,\ (E_{R\,6-1})^3,\ (E_{R\,7-1})^3,\ (E_{R\,8-1})^3,\ (E_{R\,9-1})^3,\\
&(E_{R\,3-2})^3,\ (E_{R\,4-2})^3,\ (E_{R\,5-2})^3,\ (E_{R\,6-2})^3,\ (E_{R\,7-2})^3,\ (E_{R\,8-2})^3,\ (E_{R\,9-2})^3,\ (E_{R\,4-3})^3,\\
&(E_{R\,5-3})^3,\ (E_{R\,6-3})^3,\ (E_{R\,7-3})^3,\ (E_{R\,8-3})^3,\ (E_{R\,9-3})^3\} \text{ - for the complete matrix in the}\\
&\text{root node;}
\end{aligned}$$

where (A) is a production aggregating two submatrices into one complete matrix.

Actions from E1 can be run concurrently with actions from E2. Any action from E3 can be executed only after all actions from E1 and E2 have finished. Dependency relation D^1 within E1 is defined as a transitive closure of the following set:

$$\begin{aligned}
D^1 = \{&\{(E_{R\,2-1})^1, (E_{R\,3-2})^1\},\ \{(E_{R\,3-1})^1, (E_{R\,3-2})^1\},\ \{(E_{R\,2-1})^1, (E_{R\,4-2})^1\},\ \{(E_{R\,4-1})^1, (E_{R\,4-2})^1\},\\
&\{(E_{R\,2-1})^1, (E_{R\,5-2})^1\},\ \{(E_{R\,5-1})^1, (E_{R\,5-2})^1\},\ \{(E_{R\,2-1})^1, (E_{R\,6-2})^1\},\ \{(E_{R\,6-1})^1, (E_{R\,6-2})^1\},\\
&\{(E_{R\,3-2})^1, (E_{R\,4-3})^1\},\ \{(E_{R\,4-2})^1, (E_{R\,4-3})^1\},\ \{(E_{R\,3-2})^1, (E_{R\,5-3})^1\},\ \{(E_{R\,5-2})^1, (E_{R\,5-3})^1\},\\
&\{(E_{R\,3-2})^1, (E_{R\,6-3})^1\},\ \{(E_{R\,6-2})^1, (E_{R\,6-3})^1\}\} \cup \{\text{each action is dependent on itself}\}
\end{aligned}$$

Dependency relation D^2 within E2 is defined analogously to D^1. D^3 within E3 contains analogous actions to those in

E1 and E2, extended from 6 rows in the child submatrices to 9 rows in the complete matrix. Additionally, each action in E3 depends on (A).

The developed model allows for the derivation of the Foata Normal Form describing the concurrent execution of the two dimensional version of the solver. All tasks from a single class can be executed in concurrent.

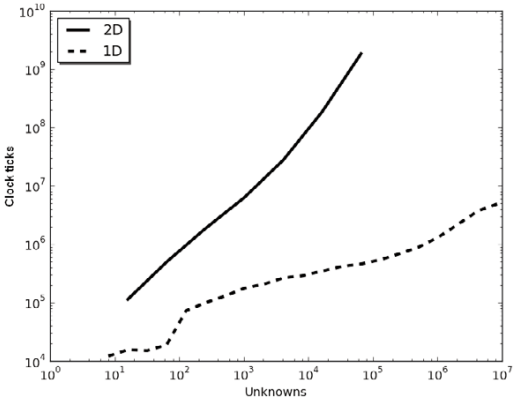

Figure 11: Execution time measured in clock ticks for 1D and 2D solver.

4. Numerical experiments

We conclude the presentation with two numerical experiments performed with one and two dimensional version of the solver. The experiments were performed on NVIDIA GeForce 8800gt graphic card, with 14 multiprocessors, with 8 cores per the multiprocessor. The blocks of tasks associated with the Foata Normal Form are scheduled into multiprocessors. If all tasks correspond to the same graph grammar production, the multiprocessor can execute all these tasks fully in parallel. The execution of tasks is interchange with global synchronization barriers, according to the Foata Normal Form. The experiments are summarized in Figure 11. The horizontal axis denotes the number of unknowns n defined as follows. For the one dimensional case the number of unknowns is equal to the number of leaves in the elimination tree, while for the two dimensional case the number of unknowns is equal to the number of leaves times the number of nodes in a vertical slice of the mesh assigned to a single leaf. The computational cost for the parallel algorithm is equal to the number of classes in the Foata Normal Form. There are several classes for the generation of the elimination tree, for the elimination over the tree and the backward substitution. The elimination tree can be constructed in $log\,(n)$ steps, in both one and two dimensional cases (where all the logarithms considered here have basis equal to 2). For the one dimensional case the eliminations and backward substitutions can be performed in $log\,(n)$ steps. Thus, the computational cost for the one dimensional case is equal to $O\,(log\,(n))$. When the number of leaves n is greater than the number of processors p, the execution time must be multiplied by $\frac{n}{p}$. This corresponds to the logarithmic shape of the 1D curve in Figure 11. For the two dimensional case, the eliminations and backward substitutions can be performed in $nlog\,(n)$ steps, since there are $log\,(n)$ levels in the elimination tree, and each node contains now $2n^{1/2}$ unknowns, and eliminates (or solves in backward substitution stage) $n^{1/2}$ unknowns, except the root node, where we have $3n^{1/2}$ unknowns (all unknowns to be eliminated). Each node (except the root node) can be processed in $2n^{1/2} * n^{1/2} = O\,(n)$ operations, since different rows are processed in concurrent. The root node requires $\left(3n^{1/2}\right)^2 = O\,(n)$ operations. The total execution time for the 2D case is then $O\,(nlog\,(n))$. Again, when the number of leaves n is greater than the number of processors p, the execution time must be multiplied by $\frac{n}{p}$. This corresponds to the shape of the 2D curve in Figure 11.

5. Conclusions

We developed the graph grammar based parallel multi-frontal multi-thread direct solver. The solver algorithm was expressed as a sequence of graph grammar productions. The alphabet and the dependency relation was identified for the sequence of productions modeling the execution of the solver algorithm. This allowed for construction of the scheduler for shared memory architecture parallel machine, based on the Foata Normal Form. Thus, the execution of the solver consists of several steps, where independent tasks are executed in concurrent, interchanged with the synchronization barriers. The solver was implemented and tested on NVIDIA Cuda graphic card. The solver was applied for the simple banchmark heat transfer problem, however it can be generalized to the class of elliptic equations, discretized by either finite difference or finite element method. The future work will involve the generalization of the solver for Finite Element Method and testing the solver algorithm on more challenging problems over a cluster of graphic cards.

Acknowledgments

The work reported in this paper was supported by the Polish MNiSW grants no. NN 519 405737 and NN 519 318635.

References

[1] Diekert V., Rozenberg G., The book of traces, World Scientific Publishing (1995).
[2] Grabska E., Theoretical concepts of graphical modeling. Part One: Realization of CP-graphs, Machine Graphics and Vision 2, 1 (1993) p. 3–38.
[3] Grabska E., Theoretical concepts of graphical modeling. Part Two: CP-graph grammars and languages, Machine Graphics and Vision, 2, 2 (1993) p. 149–178.
[4] Grabska E, Hliniak G., Structural aspects of CP-graph languages, Schedae Informaticae, 5 (1993) p. 81–100.
[5] Duff I. S., Reid J. K., The multifrontal solution of indefinite sparse symmetric linear systems. ACM Trans. on Math. Soft., 9 (1983) 302-325.
[6] Khaira M. S., Miller G. L., Sheffler T. J., Nested Dissection: A survey and comparison of various nested dissection algorithms, CMU-CS-92-106R, Computer Science Department, Carnegie Mellon University (1992).
[7] Paszyński M., On the parallelization of self-adaptive *hp*-Finite Element Method. Part I. Composite Programmable Graph Grammar Model, Part II. Partitioning Communication Agglomeration Mapping Analysis, Fundamenta Informaticae 43(9) (2009) 411-457.
[8] Paszyński M., R. Schaefer, Graph Grammar Driven Parallel Partial Differential Equation Solver, Concurrency & Computations, Practise & Experience (2010) in press, DOI:10.1002/cpe.1533.