Застосування Алгоритму Імітації Відпалу для Розподілення Даних при Паралельному Моделюванні Неізотермічного Напружено-Деформованого Стану з Урахуванням Фільтрації

В.О. Богаєнко
Інститут кібернетики ім. В.М. Глушкова НАН України
Київ, Україна
sevab@ukr.net

Application of Simulated Annealing Partitioning for Parallel Non-Isothermal Filtration-Affected Stress-Strain State Modelling

V.O. Bohaienko
V.M. Glushkov Institute of cybernetics of NAS of Ukraine
Kyiv, Ukraine
sevab@ukr.net

***Анотація*—У роботі розглядаються схеми розподілу даних в межах паралельного алгоритму скінчено-елементного моделювання напружено-деформованого стану неукріплених стінок дренажних каналів у неізотермічних умовах з урахуванням фільтраційних процесів. Для статичного та динамічного розроділу елементів сітки були застосовані алгоритм імітації відпалу та алгоритм декомпозиції на основі зменшення півширини стрічки. Алгоритм імітації відпалу був модифікований за допомогої евристики, аналогічної застосовуваній у алгоритмах мурашиних колоній. Приведені результати тестування ефективності цих алгоритмів.**

***Abstract*—Data partitioning schemes within parallel implementation of finite element modelling of non-isothermal filtration-affected stress-strain state of drainage channel walls are considered in the paper. Simulated annealing and reduced bandwidth decomposition algorithms are used for static and dynamic partitioning of a mesh. Heuristic, similar to the one used in ant colony algorithms, is proposed for simulated annealing algorithm. Algorithms efficiency testing results are presented.**

Ключові слова—напружено-деформований стан; неізотермічність; фільтрація; метод скінчених елементів; паралельні алгоритми; розбиття сітки; алгоритм імітації відпалу

Keywords—stress-strain state; non-isothermal; filtration; finite elements method; parallel algorithms; mesh partitioning; simulated annealing

I. Introduction

High computation complexity of numerical methods for modelling coupled processes in soils makes urgent a development of parallel algorithms for distributed memory systems. When differential equations system is discretized using finite element method, partitioning of the mesh used to map its elements onto the network of processes highly influence parallel computations performance.

Several approaches to solve optimal mesh partitioning problem are known. The first is recursive bisection which focuses on splitting nodes graph (e.g., spectral graph bisection [1, 2]) or mesh geometry (e.g., recursive coordinate bisection [3]). Such methods are successfully used for initial static mesh partitioning. Diffusive algorithms [4, 5] transfer mesh nodes from one process to another using information about their performance and are widely used for dynamic repartitioning. Another efficient approach for this purpose is an usage of simulated annealing as a solver of combinatorial optimization problem [6, 7]. Greedy [8] and heuristic [9] algorithms are also developed and used. Reduced bandwidth decomposition algorithm [10, 11] can be used for both initial partitioning and repartitioning. This algorithm renumbers mesh nodes with one of bandwidth minimization algorithms and assigns equally

sized blocks of nodes with sequential numbers to each process. Some of researches [12-14] focus on using swarm intelligence algorithms for solving problems of task allocation in general and finite element mesh partitioning particularly.

Usage of simulated annealing partitioning with reduced bandwidth decomposition is studied in the paper to accelerate parallel finite element analysis of stress-strain state dynamic of non-reinforced walls of drainage channels. Ant colony-like heuristics is proposed to increase efficiency of simulated annealing.

II. Problem statement

Modelling of non-isothermal stress-strain state of a soil is considered in the paper taking filtration into account. Mathematical model of the process consists of dynamic filtration, heat transfer, and stress-strain state equations as follows [15]:

$$\tilde{\mu}\frac{\partial h}{\partial t}-\frac{\partial}{\partial x}\left(K_f(T,P)\frac{\partial h}{\partial x}\right)-\frac{\partial}{\partial y}\left(K_f(T,P)\frac{\partial h}{\partial y}\right)=0,$$

$$c_T\frac{\partial T}{\partial t}-\frac{\partial}{\partial x}\left(\lambda_T\frac{\partial T}{\partial x}-c_w v_x T\right)-\frac{\partial}{\partial y}\left(\lambda_T\frac{\partial T}{\partial y}-c_w v_y T\right)=0,$$

$$\rho_s\frac{\partial^2 u}{\partial t^2}-\mu\Delta u-(\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)-\frac{\partial P}{\partial x}=0, \qquad (1)$$

$$\rho_s\frac{\partial^2 v}{\partial t^2}-\mu\Delta v-(\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)-\frac{\partial P}{\partial y}=-\rho_s g,$$

$$(x,y)\in\Omega,$$

where h is a water head, T – temperature, $w=(u,v)^T$ – soil displacement vector, $P=\rho_w g(h-y)$ – pressure, $K_f(T,P)$ – filtration coefficient, $(v_x,v_y)^T$ – filtration velocity, λ, μ – Lamé parameters, $\tilde{\mu}$ – soils water capacity, c_T, c_w – volumetric heat capacities of soil and water, λ_T – thermal conductivity coefficient, ρ_s, ρ_w – densities of soil and water.

Equation system (1) is considered in the domain Ω depicted in fig.1 that represents soil massif with two drainage canals.

The following boundary and initial conditions were set:

$$h(x,y,t)=y,\ (x,y,t)\in\Gamma^1,$$

$$h(x,y,t)=y_0+\tilde{h}_1(t),\ (x,y,t)\in CD,$$

$$\tilde{h}_1(t)=\frac{1}{|CD|}\int_{CD}\int_0^t v_y(x,y,\tau)d\tau d\Gamma,$$

$$h(x,y,t)=y_0+\tilde{h}_2(t),\ (x,y,t)\in GH, \qquad (2)$$

$$v_y(x,y,t)=-K_f(T,P)\frac{\partial h}{\partial y},\ \frac{\partial h}{\partial n}=0,\ \frac{\partial T}{\partial n}=0,\ (x,y)\in\Gamma^2,$$

$$T(x,y,t)=T_1,\ (x,y)\in\Gamma^1,\ T(x,y,t)=T_2,\ (x,y)\in OK,$$

$$u=0,\ (x,y)\in OA\cup JK,\ u=v=0,\ (x,y)\in OK,$$

$$\tau_s=0,\ \sigma_n=0,\quad (x,y)\in\Gamma^1,$$

$$\tilde{h}_2(t)=\frac{1}{|GH|}\int_{GH}\int_0^t v_y(x,y,\tau)d\tau d\Gamma,$$

$$h(x,y,0)=H_0(x,y),\quad T(x,y,0)=T_0(x,y),$$
$$u(x,y,0)=U_0(x,y),$$
$$v(x,y,0)=V_0(x,y),\quad \frac{\partial u}{\partial t}(x,y,0)=\frac{\partial v}{\partial t}(x,y,0)=0, \qquad (3)$$
$$(x,y)\in\Omega.$$

where

$$\Gamma^1=AB\cup BB'\cup E'E\cup EF\cup FF'\cup I'I\cup IJ,$$

$\Gamma^2=OA\cup OK\cup JK$, σ_n, τ_s are normal and tangent components of stress vector.

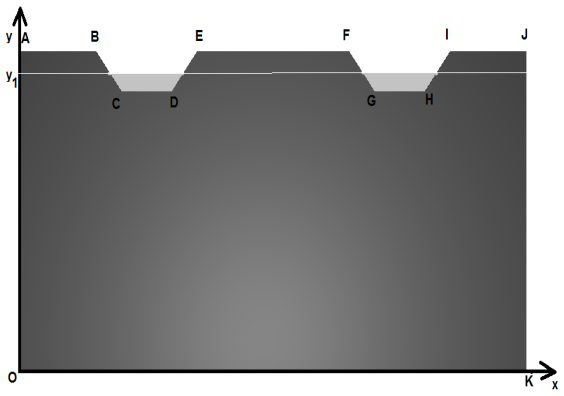

Fig.1. Solution domain

Problem (1)-(3) was numerically solved using finite element discretization on space variables and Crank-Nicolson discretization on time variable by the scheme similar to described in [16].

III. Parallel algorithms features

Solution of the discrete problem on distributed memory systems is considered on the base of the approach described in [16, 17]. Linear systems matrix is stored in compressed row storage format. Parallel version of BiCGStab [18] was used to solve the linear system with multithreaded execution within each process. Complete finite element mesh was stored in the memory of each process.

Performance of the parallel algorithm depends on uniformity of nodes distribution over the network of processors and time spent on data exchanges. The latter itself depends on the volume of data and the number of exchange operations. Partitioning of finite element mesh has to be done statically, without knowledge about computational system performance, at the start of solution process while dynamic repartitioning can be done further.

Considered problem is solved over simple domains that describe soil massifs. On the other hand, it takes into consideration several coupled physical processes that leads to linear systems with ill-conditioned matrices of high density. In such situation, different uniform data distributions can be simply reached and minimization of data exchange time along with dynamic adaptation of mesh distribution on the performance of the system plays an essential role.

We consider two variants of linear system matrix organization:

– with rows and columns grouped by functions having a matrix of 6x6 blocks;

– with rows and columns grouped by mesh nodes;

In the first case, rows of the matrix are initially distributed among processes the way to make each processes working time equal. Each row of the matrix is given a weight equal to average number of operations needed to compute one non-zero entry in this row multiplied by the number of non-zero elements. Sequential blocks of rows are assigned to processes such that a sum of row weights in the block of each process is equal. In the process of computations, as each processes performance can be measures, dynamic matrix repartitioning is done once per a given number of time steps. While doing repartitioning, weights of rows are multiplied by average time spent by owning process to make computations associated with this row.

In the second case, as the problem is solved on relatively simple meshes and for long time intervals, usage of the simplest algorithm for initial partitioning with probabilistic dynamic repartitioning can be efficient. We consider the partitioning scheme where initial partitioning is done by reduced bandwidth decomposition that uses Cuthill-McKee algorithm [19] and dynamic repartitioning done by modified simulated annealing method.

IV. Simulated annealing partitioning

Simulated annealing is a general probabilistic optimization method that simulates physical system cooling. It minimizes goal function F associated with systems state T called 'temperature' by proposing and then accepting or rejecting changes to systems state. The change is accepted unconditionally if it lowers goal function value. Otherwise, it is accepted with probability $e^{-k(F_1-F_0)/T}$, where F_1 - goal function values in changed state, F_0 - goal function value in initial state, k - given constant. Algorithm ends search when a given number of sequentially rejected changes or a given total number of changes are reached.

For mesh partitioning, system state change is done by random selection of a node which is then reassigned from current owner to random process. Several heuristics are used to make algorithm more efficient:

[1] only node on boundaries of node blocks owned by different processes can be selected for changing;

[2] node can be transferred only to the process that owns one of the neighbouring nodes;

[3] as it can be more efficient to change assignment of a cluster of connected nodes owned by the same process than of a single node, such clusters are generated with given probability of node addition;

We propose a modification of the algorithm to make use of information about nodes where changes were successful. For this purpose, approach used in ant colony algorithm [20] can be applied. Ant colony algorithm simulates ants' behaviour: an ant searching for food spreads pheromone that is used by other ants to do further search in 'efficient' areas. Pheromone is evaporating over time. For the considered algorithm such heuristic can be used the following way:

[4] pheromone value is associated with every node;

[5] nodes, neighbouring to nodes of changed cluster, have pheromone value of $m_f(F1-F0)$ added, where m_f - given multiplier;

[6] pheromone values for every node are divided by given constant d_f after each successful change;

[7] node selection is done weighted-randomly with pheromone values as weights;

We use the following form of goal function:

$$F = k_1 \max_p N_p + k_2 \max_p C_p + k_2 \max_p E_p, \qquad (4)$$

where N_p - number of nodes owned by process p, C_p - number of nodes, own by process p, that have neighbours owned by other processes, E_p - number of processes with which process p have communications, k_1, k_2, k_3 - given coefficients that depend on computational system and network performance. Based on the form of goal function, the following additional heuristics were used:

[8] weight used for selecting node for changing is multiplied by $k_1 N_p + k_2 C_p + k_2 E_p$, where p is a process that owns the node. Such additional weight multiplier makes more probable selection of nodes owned by processes that highly influence goal function value;

[9] if a change of nodes' ownership do not change a value of goal function, additional check is done: change is accepted if $\max_p E_p$ lowers, where E_p - number of edges which connect a node owned by process p and a node owned by other process.

V. Parallel algorithms testing

Several numerical experiments were carried out to test an efficiency of considered algorithms and their proposed modifications on SKIT-3 cluster of Institute of cybernetics of NAS of Ukraine on 4-cored nodes.

For testing purposes, problem (1)-(3) was being solved with the following values of parameters:

[10] $K_f(T,P) = \overline{K}_f(10)(0.7 + 0.03T)$;

[11] $\overline{\overline{K}}_f(10) = \begin{cases} \overline{\overline{K}}f, P \geq 0, \\ \overline{\overline{K}}_f e^{2.48P}, P < 0, \end{cases} \quad \overline{\overline{K}}_f = 0.3\, m/day$;

[12] $\rho_s = 1940\, kg/m^3$, $\tilde{\mu} = 10^{-3}$;

[13] Young modulus of soil - $E = 5 \cdot 10^6\ kg/m^3$;

[14] Poisson coefficient of soil - $\nu = 0.3$.

In the first series of experiments, parallel algorithms that uses per function and per node row blocking in linear systems matrices were tested for their performance. Two variants of mapping onto computational system were considered: the first with one thread per process and four processes per node, the second – with one process per node and four threads per process. Initial mesh nodes distribution over processes was computed using reduced bandwidth decomposition approach with Cuthill-McKee numbering. Above described weight coefficients were used in the case of per function row blocking. Obtained algorithms speed-up depending on total number of threads is given in fig.2.

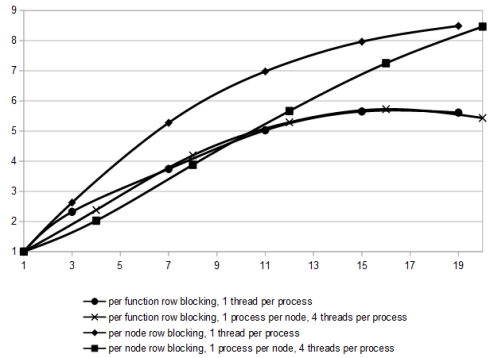

Fig. 2. Parallel algorithms speedup

The results show that per node row blocking leads to better performance because it significantly (up to 5 times) lowers a volume of data exchanges comparing with per function blocking. Use of multithreading also lowers time spent on exchanges but makes linear system matrix filling slower. The latter can be explained by less efficient memory cache utilization. In the conducted experiments, scheme without use of multithreading was faster in most cases.

In the second series of experiments simulated annealing partitioning response on parameter values was studied comparing obtained values of goal function (4) with $k_1 = k_1 = 1$, $k_3 = 0$. Number of iterations done by the algorithm was limited by 10000. As the algorithm is probabilistic, series of 20 runs were done and minimal values of goal function were averaged for further analysis. Standard deviation in all series of experiments was not more than 10% of average minimal goal function value.

Highest efficiency in the case of partitioning the mesh to map onto 15 processes and initial distribution given by reduced bandwidth decomposition algorithm was obtained for $m_f = 10$, $d_f = 1.01$, $k = 100$. Such initial distribution is close to the minimal possible, so efficiency of partitioning optimization in the terms of goal function value minimization was relatively small and was equal to 15%. When ant colony-like heuristics was not used, only 4% efficiency was obtained.

Efficiency of the algorithm in the case of obtained optimal parameter set for random and predefined initial distributions depending on number of processes to map onto is presented on fig.3.

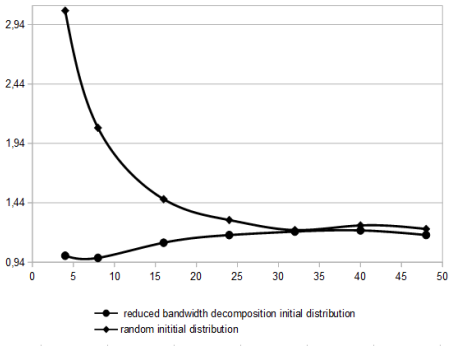

Fig. 3. Efficiency of simulated annealing partitioning

The results show that efficiency lowers when number of processes becomes higher for random initial distribution case. When initial distribution is generated by reduced bandwidth decomposition algorithm, efficiency, in contrary, grows. However, in this case, algorithm lowers initial goal function value only for number of processes more than 15.

When applying considered mesh partitioning scheme for solving problem (1)-(3) on specific computational system, initial guess of its performance parameters is crucial. Series of experiments for testing simulated annealing partitioning as static method to partition a grid were conducted and times spent on calculations and data exchanges per one iteration of BiCGstab were compared for the case of initial distribution generated by reduced bandwidth decomposition algorithm and the case when it is optimized by proposed modification of simulated annealing with $m_f = 10$, $d_f = 1.01$, $k = 250$, $k_1 = 1$, $k_2 = 0.175$, $k_3 = 10$ and maximal number of iteration equal to 30000. In situations when up to 56 cores were used, only in several cases up to 6% speed-up was obtained.

Regarding low efficiency of considered algorithms as static partitioners, the last series of numerical experiments were carried out to study computational systems performance coefficients assessment and further repartitioning influence on time spent on problem solution. Repartitioning was conducted on each 10-th iteration. Computational systems performance coefficients were calculated using averaged measures done while finding solution on each time step. For simulated annealing algorithm, maximal number of iterations was set to 1/10 of initial when doing repartitioning. Speed-up was calculated as time spent per one BiCGstab iteration in the case of non-optimized partitioning divided by the same time spent for solution on optimized mesh partition. Obtained results for speed-up depending on number of repartitions for different number of processes are depicted on fig.4 for per function row blocking and on fig.5 for per node row blocking and

application of the proposed modification of simulated annealing algorithm.

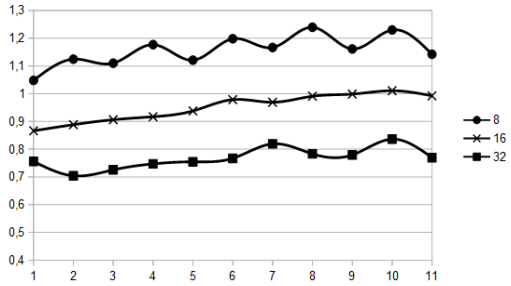

Fig. 4. Speed-up while using dynamic repartitioning with per function matrix rows blocking

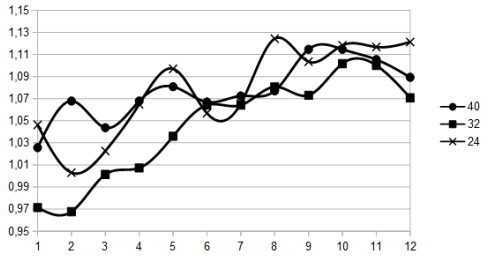

Fig. 5. Speed-up while using dynamic repartitioning with per node matrix rows blocking

Algorithm for dynamic repartitioning in the case of per function row blocking implicitly takes time spent on data exchanges into account. It increases the volume of data to be exchanged on increase of the number of processes becoming inefficient when it grows more than 16. However it uniforms time spent on computations by different processes giving ~20% speedup in the experiment with 8 processes.

In the case of per node row blocking, efficiency of dynamic repartitioning slightly depend on number of processes. Simulated annealing algorithm has given performance speed-up up to ~13% in all experiments focusing on minimization of volume and especially number of data exchange.

VI. Conclusions

Considered problem is solved in rather simple domains making application of even the simplest mesh partitioning algorithms efficient while doing parallel computations. However, optimization of such partitioning is possible both statically and dynamically. It was shown that per node row blocking in linear systems matrix with reduced bandwidth decomposition partitioning is more efficient for the considered problem comparing with the similar decomposition and per function row blocking. Simulated annealing partitioning modified by ant colony-like heuristic was used to further optimize the partition after reduced bandwidth decomposition algorithm. While giving no or negligible performance speed-up when used as static partitioner, simulated annealing produce ~13% speed-up when used for dynamic repartitioning taking computations systems performance coefficients estimations into account.

Considering a significant influence of the simulated annealing algorithm parameters on its performance, further research can be done to study efficiency of heuristic algorithms, e.g. neural networks, usage for finding their optimal values.

References

[1] H. Simon, “Partitioning of un-structured problems for parallel processing” in Computing Systems in Engineering, vol. 2, no. 2/3, 1991, pp. 135-148.

[2] A. Pothen et al., “Partitioning sparse matrices with eigenvectors of graphs” in SIAM J. Matrix Anal., 11, 1990, pp. 430–452.

[3] M.J. Berger and, S.H. Bokhari, “A partitioning strategy for nonuniform problems on multiprocessors” in IEEE Trans. Computers, 36, 1987, pp. 570–580.

[4] G. Cybenko, “Dynamic load balancing for distributed memory multiprocessors” in J. Parallel Distrib. Comput., 7, 1989, pp. 279–301.

[5] H. Meyerhenke at al, "A new diffusion-based multilevel algorithm for computing graph partitions of very high quality" in IEEE International Symposium on Parallel and Distributed Processing 2008, 2008, pp.1-13.

[6] J. Chen and V.E. Taylor, “Mesh partitioning for efficient use of distributed systems” in IEEE Transactions on Parallel and Distributed Systems, 1, 2002, pp. 67–79.

[7] R.D. Williams, “Performance of dynamic load balancing algorithms for unstructured mesh calculations” in Concurrency: Pract. Exper., 3, 1991, pp. 457–481.

[8] C. Farhat, “A simple and efficient automatic FEM domain decomposer” in Computer & Structures, vol. 28, 5, 1988, pp. 579-602.

[9] B.W. Kernighan and S. Lin, “An efficient heuristic procedure for partitioning graphs” in The Bell System Technical Journal, 49, 1970, pp. 291-307.

[10] J.G. Malone, “Automatic mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers” in Comput. Methods Appl. Mech. Eng., 70, 1988, pp.27-58.

[11] R.J. Collins, “Bandwidth reduction by automatic renumbering” in Int. j. numer. method eng., 6, 1973, pp. 345-356.

[12] A.I. Khan and B.H.V. Topping, “Subdomain generation for parallel finite element analysis” in Computing Systems in Engineering, vol. 4, issues 4–6, 1993, pp. 473-488.

[13] N. Chmait and K. Challita, “Using Simulated Annealing and Ant-Colony Optimization Algorithms to Solve the Scheduling Problem” in Computer Science and Information Technology, 1(3), 2013, pp.208-224.

[14] A.J. Page et al., “Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system” in J. Parallel Distrib. Comput. 70, 2010, pp. 758-766.

[15] V.A. Bohaienko et al., “Analysis of numerical modelling of non-isothermal processes in soil massif (in Russian)” in Kompiuternaia matematica, №2, 2016, pp.2-13.

[16] V.A. Bohaienko et al., “Analysis of numerical modelling of soil massif dynamic in the case of non-steady state pressure filtration (in Russian)” in USiM, №4, 2014, pp.33-40.

[17] V.A. Bohaienko, “Parallel algorithm for modelling dynamic consolidation process in soil media of layered structure (in Russian)” in International Conference “Parallel and Distributed Computing Systems PDCS 2013” Collection of scientific papers, 2013, pp. 58-62.

[18] Y. Saad, “Iterative methods for sparse linear systems, 2 edition” - Society for Industrial and Applied Mathematics, 2003.

[19] E. Cuthill and J. McKee, “Reducing the bandwidth of sparse symmetric matrices” in Proc. 24th Nat. Conf. ACM, 1969, pp.157–172.

[20] M. Dorigo, “Optimization, Learning and Natural Algorithms”, PhD dissertation, Politecnico di Milano, Italy, 1992