Parallel Bioinspired Algorithms in Optimization of Structures

Wacław Kuś[1] and Tadeusz Burczyński[1,2]

[1] Department for Strength of Materials and Computational Mechanics, Silesian University of Technology, ul. Konarskiego 18a, 44-100 Glwice, Poland
waclaw.kus@polsl.pl, tadeusz.burczynski@polsl.pl
[2] Institute for Computer Modelling, Cracow University of Technology ul. Warszawska 24, 31-155 Cracow, Poland

Abstract. The parallel versions of bioinspired algorithms are presented in the paper. The parallel evolutionary algorithms and artificial immune systems are described. The applications of bioinspired algorithms to optimization of mechanical structures are shown. The numerical tests presented in the paper were computed with use of grid based on Alchemi framework.

1 Introduction

The optimization methods inspired by biological mechanisms have become very popular in last few decades. Most of them give good results optimizing multimodal functions. The paper describes the computational intelligence algorithms: evolutionary algorithms and artificial immune system. The optimization of mechanical structures is a long time process because of hundreds or even thousands of objective function evaluations. The objective function evaluation is connected with a direct problem which solved by means of FEM [12] in most cases. The wall computation time can be shorten when parallel algorithms are used [3][2]. The computational grids provide sophisticated and user friendly environment for performing time consuming computations. The grids give opportunity to use distributed resources in engineering optimization problems [6]. The Alchemi framework [1] is used in the paper. The shape optimization of the anvils in two-stage forging process is considered as a benchmark problem.

2 Grid Based on Alchemi Framework

The Alchemi framework[1] was used to construct a grid. The Alchemi framework is based on Windows .NET. This makes Alchemi useful only on hardware using Windows operating system. The Alchemi consists of few elements: Alchemi Manager - the central host with scheduling capabilities, one manager is needed for grid or part of grid, Alchemi Executors - the hosts performing computations, Alchemi Cross Platform Manager - web services based manager with ability to communicate with non-Alchemi parts of the grid. The Alchemi Manager host

R. Wyrzykowski et al. (Eds.): PPAM 2007, LNCS 4967, pp. 1285–1292, 2008.

can also be used as a Executor one. The security policy is based on usernames and passwords. The users can be grouped. The end-users, executors and administrator groups are available by default. The information about users, tasks, jobs, executor hosts are stored in database connected with Alchemi Manager. The communication between Alchemi Manager and Executors are performed using TCP/IP selected ports. These ports need to be available through firewalls. The architecture of the grid based on Alchemi is shown in Fig. 1[1].

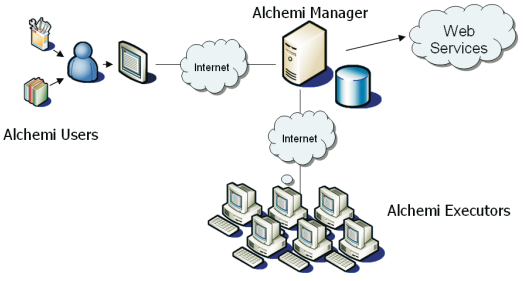

Fig. 1. Architecture of grid based on Alchemi framework[1]

The most important advantage of Alchemi framework is API provided for grid applications developers. The execution of grid applications is performed using remote threads running on executors. The communication between remote threads is prohibited. The task submitted to grid consists of threads. The Alchemi framework contains also Alchemi Console for monitoring tasks, threads and executors. The Alchemi Manager and Executor can work as a system service.

3 Evolutionary Algorithms

The genetic and evolutionary algorithms (EA) [8] are based on mechanisms taken from biological evolution of species. The selection based on the individual fitness, mutations in chromosomes and individuals crossover are adopted. The genetic algorithms operate on binary coded chromosomes. The term evolutionary algorithm is more widely used for different modifications of genetic algorithms (also for algorithms operating on genes containing floating point numbers). The evolutionary algorithms operate on a population of individuals. The individuals contain one chromosome in most cases. The following description concerns the evolutionary algorithm used in numerical examples. The staring population is created randomly. Next the fitness function values are computed for each chromosome. The selection chooses chromosomes for a new parent subpopulation taking into account fitness function values. Evolutionary operators change chromosomes' genes and create chromosomes for the offspring population. The uniform and Gaussian mutations and the simple crossover are randomly chosen to perform chromosome changes. The new chromosome are evaluated. The

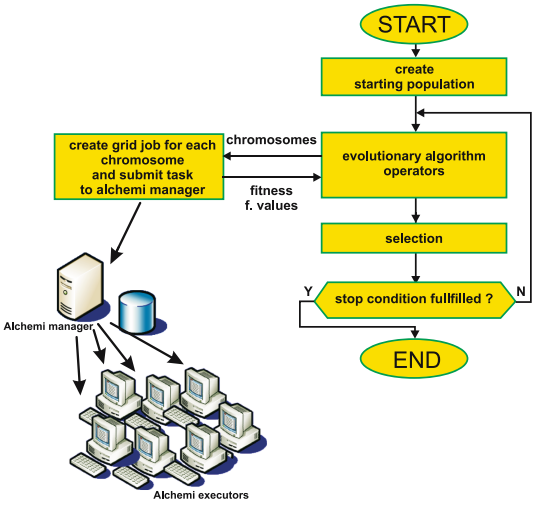

Fig. 2. The flowchart of the evolutionary algorithm

algorithm works iteratively till the end optimization condition is fulfilled. The flowchart of evolutionary algorithm is presented in Fig. 2[7].

4 Artificial Immune Systems

The artificial immune systems are developed on the basis of mechanism discovered in biological immune systems. An immune system is a complex system which contains distributed groups of specialized cells and organs. The main purpose of the immune system is to recognize and destroy pathogens - funguses, viruses, bacteria and improper functioning cells. The lymphocytes cells play a very important role in the immune system. The lymphocytes are divided into several groups of cells. There are two main groups B and T cells, both contains some subgroups (like B-T dependent or B-T independent). The B cells contains antibodies, which could neutralize pathogens and are also used to recognize pathogens. There is a big diversity between antibodies of the B cells, allowing recognition and neutralization of many different pathogens. The B cells are produced in bone marrow in long bones. A B cell undergoes a mutation process to achieve big diversity of antibodies. The T cells mature in thymus, only T cells recognizing non self cells are released to the lymphatic and the blood systems. There are also other cells like macrophages with presenting properties, the pathogens are processed by a cell and presented by using MHC (Major Histocompatibility Complex) proteins. The recognition of a pathogen is performed in a few steps. First, the B cells or macrophages present pathogen to a T cell using MHC, the T cell decides if the presented antigen is a pathogen. The T cell gives a chemical signal to B cells to release antibodies. A part of stimulated B cells goes to a lymph node and proliferate (clone). A part of the B

cells changes into memory cells, the rest of them secrete antibodies into blood. The secondary response of the immunology system in the presence of known pathogens is faster because of memory cells. The memory cells created during primary response, proliferate and the antibodies are secreted to blood. The antibodies bind to pathogens and neutralizes them. Other cells like macrophages destroy pathogens. The number of lymphocytes in the organism changes, while the presence of pathogens increases, but after attacks a part of the lymphocytes is removed from the organism. The artificial immune systems (AIS) [5] take only few elements from the biological immune systems. The most frequently used are the mutation of the B cells, proliferation, memory cells, and recognition by using the B and T cells. The artificial immune systems have been used to optimization problems, classification and also computer viruses recognition. The cloning algorithm Clonalg presented by von Zuben and de Castro [4] uses some mechanisms similar to biological immune systems to global optimization problems. The unknown global optimum is the searched pathogen. The memory cells contain project variables and proliferate during the optimization process. The B cells created from memory cells undergo mutation. The B cells are evaluated and better ones exchange memory cells. In Wierzchoń [11] version of Clonalg the crowding mechanism is used - the diverse between memory cells is forced. A new memory cell is randomly created and substitutes the old one, if two memory cells have similar project variables. The crowding mechanism allows finding not only the global optimum but also other local ones. The presented approach is based on the algorithm presented in [11]. The mutation operator is changed. The Gaussian mutation is used instead of nonuniform mutation in the presented approach. The parallel artificial immune system (PAIS) was introduced by [10] for classification problems. An artificial immune system is implemented as one master process, other processes - workers evaluate objective functions for B cells. The memory cells are created randomly. They proliferate and mutate creating B cells. The number of clones created by each memory cell is determined by the memory cells objective function value. The objective functions for B cells are evaluated. The selection process exchanges some memory cells for better B cells. The selection is performed on the basis of the geometrical distance between each memory cell and B cells (measured by using design variables). The crowding mechanism removes similar memory cells. The similarity is also determined as the geometrical distance between memory cells. The process is iteratively repeated until the stop condition is fulfilled. The stop condition can be expressed as the maximum number of iterations. The master part of the algorithm is placed in the central processors and workers communicate only with the master. The Alchemi grid environment was used in the computations. The flowchart of the artificial immune system is shown in Fig. 3.

5 Numerical Example

The shape optimization of anvils in first stage of two-stage forming is considered as a numerical example [7]. The forging is performed using flat die in the

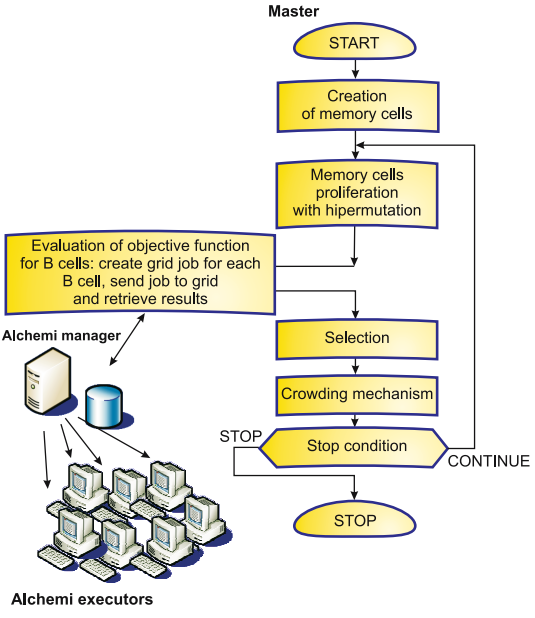

Fig. 3. The flowchart of the artificial immune system

second stage. The goal of the optimization is to find such shape of anvils which gives cylindrical product of forging after second stage. The product after forging have barreled shape. The product should have cylindrical shape after two-stage forging. The forging is simulated using MSC.Marc[9] program. The finite element method is used during direct problem solution. The preform is made from highly nonlinear material. The material nonlinearities and different shapes of the preform have influence on fitness function computing time. The anvils were described using 8 parameters of the control polygon of NURBS curve. The objective function was defined as a difference between ideal cylindrical shape and shape obtained after forging [7].

The optimization were performed by using evolutionary algorithm and artificial immune system. The results obtained using both algorithms were very close to each other.

The results of two-stage forging when only flat anvils were used are presented in Fig. 4. The barreling of the product can be easily observed. The results after optimization using both algorithms are presented in Fig. 5. The shape of the product is very close to the cylindrical one.

The theoretical speedup of parallel population-based algorithms can be very close to the linear speedup. The speedup depends on number of chromosomes, B-cells (number of objective functions to be evaluated in each iteration) and number of processors. Consider the number of individuals n and number of processors n_p. We assume that time need to compute fitness function value is constant and equal to t_f. The time need to perform communications between

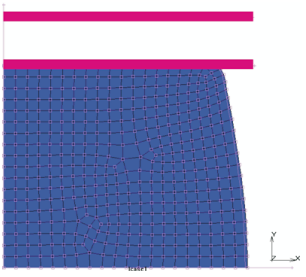

Fig. 4. The product after forging using flat anvils

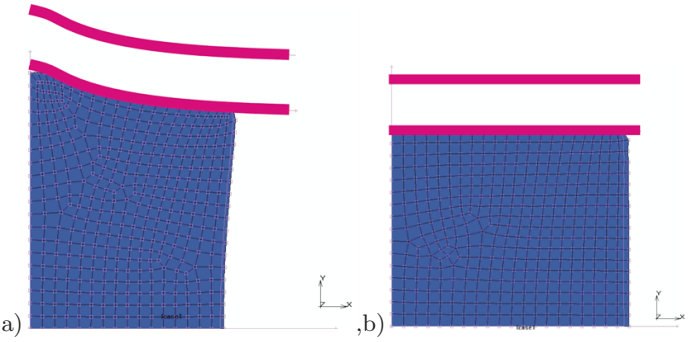

Fig. 5. The product after forging by using the best found anvils: a) after first stage, b) after second stage of forging

processors and time need to use evolutionary operators and selection is very small (comparing to the time needed for fitness function evaluation t_f) and can be neglected. The wall time of computation of one iteration of optimization algorithm using one processor is equal to $t_1 = t_f \cdot n$. When the n_p processors are used the wall time is equal to

$$t_{np} = t_f(n \backslash n_p + r) \tag{1}$$

where $\backslash$ denotes integer division (the fractional part of the result is abandoned), and the $r = 0$ is equal to:

$$r = \begin{cases} 0 \text{ if} n \bmod n_p = 0 \\ 1 \quad otherwise \end{cases} \tag{2}$$

The speedup is equal to:

$$s = \frac{t_1}{t_{np}} = \frac{n}{n \backslash n_p + r} \tag{3}$$

The speedup is equal to n_p for the case when $n_p > n$. The ideal situation is when the remainder of integer division of number of chromosomes and processors

is equal to zero, than the theoretical speedup can achieve linear speedup values $s = n_p$ for $n_p \leq n$.

The experimental measurements of speedups for different number of processors were performed. The results are shown in Tab. 1. The 18 objective functions were computed in each iteration. The theoretical measurements were performed using average objective function computation time. The results shows better measured speedups than theoretical ones in some cases. The times needed for computations of objective functions for different individuals in real experiment differs from each other. It occurs that a processor can compute two individuals when other compute only one in some cases. This situation occurs rarely, but allows to obtain and explain better performance of real experiment than theoretically predicted.

Table 1. Theoretical and measured speedups

number of processors	measured speedup	theoretical speedup
1	1.00	1.00
2	1.98	2.00
4	3.67	3.60
8	6.09	6.00
16	9.35	9.00
18	14.01	18.00

6 Conclusions

The presented parallel evolutionary algorithm and artificial immune system working in grid environment based on Alchemi framework can be successfully used in optimization problems. The complicated problems like optimization of anvils in two-stage forging can be solved using presented approaches. The time costs of optimization using both algorithms are similar. The theoretical speedup computations and comparisons with experiment were shown in the paper.

Acknowledgement

The research is financed from the Polish science budget resources in the years 2005-2008 as the research project.

References

1. Akshay, L., Rajkumar, B., Rajiv, R., Srikumar, V.: Peer-to-Peer Grid Computing and a.NET-based Alchemi Framework. In: Yang, L., Guo, M. (eds.) High Performance Computing: Paradigm and Infrastructure, Wiley Press, New Jersey (2005)
2. Burczyński, T., Kuś, W., Długosz, A., Orantek, P.: Optimization and defect identification using distributed evolutionary algorithms. Engineering Applications of Artificial Intelligence 17(4), 337–344 (2004)

3. Burczyński, T., Kuś, W.: Optimization of structures using distributed and parallel evolutionary algorithms. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2004. LNCS, vol. 3019, pp. 572–579. Springer, Heidelberg (2004)
4. de Castro, L.N., Von Zuben, F.J.: Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems 6(3), 239–251 (2002)
5. de Castro, L.N., Timmis, J.: Artificial Immune Systems as a Novel Soft Computing Paradigm. Soft Computing 7(8), 526–544 (2003)
6. Kuś, W., Burczyński, T.: Grid-based evolutionary optimization of structures. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, Springer, Heidelberg (2006)
7. Kuś, W.: Evolutionary optimization of forging anvils using grid based on Alchemi framework. In: Proc. 2nd IEEE International Conference on e-Science and Grid Computing eScience 2006, Amsterdam (2006)
8. Michalewicz, Z.: Genetic algorithms + data structures = evolutionary algorithms. Springer, Berlin (1996)
9. MSC.Marc, Users Guide (2002)
10. Watkins, A., Bi, X., Phadke, A.: Parallelizing an Immune-Inspired Algorithm for Efficient Pattern Recognition. Intelligent Engineering Systems through Artificial Neural Networks: Smart Engineering System Design 13, 225–230
11. Wierzchoń, S.T.: Artificial Immune Systems, theory and applications, EXIT, Warsaw (in Polish) (2001)
12. Zienkiewicz, O.C., Taylor, R.L.: The Finite Element Method. In: The Basis, Butterworth, Oxford, vol. 1-2 (2000)