14th World Congress in Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11 – 15 January 2021

F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

PARALLEL COMPUTING IN HP-OCP

Georgios Kazakis, Stefanos Sotiropoulos, Nikos Ath. Kallioras, Stavros Xynogalas, Chara. Ch. Mitropoulou, Stavros Chatzieleftheriou, Spyros Damikoukas, Pantelis Tsakalis and Nikos D. Lagaros[1]

[1] Institute of Structural Analysis & Antiseismic Research (ISAAR), Department of Structural Engineering, School of Civil Engineering, National Technical University of Athens (NTUA), 9, Heroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece
nlagaros@central.ntua.gr, www.veltion.ntua.gr

Key words: Structural optimization, Parallel computing, HP-OCP.

Abstract. The most computationally demanding part of structural design optimization is the solution of the FE equations and design of the structural model. Therefore, there is a need for the implementation of strategies that can reduce the computational cost of each iteration and thus manage to achieve the same optimized result with considerable reduction in the optimization time. High Performance Optimization Computing Platform (HP-OCP) is an optimization software developed in C# programming language by ISAAR-NTUA and OptiStructre Ltd. [1] which provides a holistic optimization approach for civil engineering structures. It combines powerful derivative-based and derivative-free optimization algorithms like the Projected Quasi-Newton (PQN), Constrained Optimization by Linear Approximation (COBYLA), Latin Hypercube (LH), Differential Evolution etc. [2] integrated with different structural analysis software's like SAP2000, ETABS & SCIA Engineer utilizing their abilities in finite element analysis and most importantly different design codes into the optimization procedure. To deal with the computational demand deriving from this coupling of optimization algorithms and commercial structural analysis software's parallel computational procedures have been implemented to HP-OCP. These procedures were tested in real world civil engineering problems and produced very good results. Parallel strategies are implemented both at the level of the optimization algorithm, by exploiting the natural parallelization features of the evolutionary algorithms, as well as at the level of the repeated structural analysis problems that are required by the optimization algorithm. The numerical tests presented demonstrate the computational advantages of the proposed parallel strategies, which become more pronounced in large-scale optimization problems.

1 INTRODUCTION

During the last three decades many numerical algorithms have been developed to meet the demands of engineering optimization problems. These algorithms can be classified in two categories, the gradient based and the derivative free ones. Mathematical programming belongs to the first category; these methods make use of local curvature information, derived from linearization of the original functions by using their derivatives with respect to the design variables at points obtained in the process of optimization, to construct an approximate model of the initial problem. Metaheuristic Optimization Algorithms (MOA) and in particular Evolutionary Computation (EC) are among the most widely used class of methods of the second category and in particular Evolutionary Programming (EP), Genetic Algorithms (GA), Evolution Strategies (ES) and Genetic Programming (GP). EC rely on analogies to natural processes and they are evolution-based systems maintaining a population of potential solutions. These systems have some selection process based on fitness of individuals and some recombination operators.

The parametric optimization combined with shape and/or topology optimization of large-scale three-dimensional skeletal structures is a computationally intensive task. In parametric optimization the aim is to minimize the weight of the structure under certain restrictions imposed by design codes when the characteristics of the cross-sections of the members are under investigation. While in the combined shape-topology-parametric optimization of a skeletal structure the aim is the minimization of the weight of the structure subject to performance constraints imposed by the design codes, where in addition to the cross-sections of the members, the shape of the structures is also under investigation along with the topology of the nodes of the skeletal structure.

When EC are adopted to perform the optimization, the solution of the finite element equations is of paramount importance since more than 95% of the total computing time is spent for the solution of the finite element equilibrium equations [3]. An important characteristic of EC that make them differ from mathematical programming methods is that in place of a single design point EC work simultaneously with a population of design points in the space of design variables. This allows for a natural implementation of the evolution procedure in parallel computer environments. The natural parallelization concept is implemented in the framework of EC by assigning to each processor a complete finite element analysis, without any need of inter-processor communication during the solution phase. On the other hand, if the number of available processors is greater than the population size, each finite element analysis can be assigned to more than one processor using an efficient domain decomposition solution method.

Several studies have been performed in the past implementing the natural parallelization concept [4,5]. Adeli and Kumar in [6] combined the natural parallelization characteristic of genetic algorithms with a preconditioned conjugate gradient method on the Connection Machine. In this study the natural parallelization characteristic of evolutionary algorithms together with parallel solution methods are employed. Optimized evolutionary algorithms are implemented in parallel computing environment where the natural parallelization features of the optimization algorithms are combined with the domain decomposition methods [7,8] for solving the structural analysis problems encountered in EC specially tailored to the particular type of problems at hand. Moreover, the efficiency of evolution strategies and genetic algorithms is investigated in treating large-scale design optimisation problems of skeletal structures.

2 STRUCTURAL OPTIMIZATION

The objective of structural optimization is to find a proper selection of the design variables in order to achieve the minimum weight of the structure under some behavioural constraints which are imposed by the design codes. Three are the main classes of structural optimization problems: parametric, shape and topology. In the case of parametric optimization, the design variables are the cross-sections of the members for skeletal structures or the thickness of plates and shells. In the case of shape optimization, the design variables are parameters that control the shape of the structure, while in topology optimization the aim is to optimize the layout or the topology of a structure by detecting and removing the low-stressed material or by changing the topology of the nodes in a skeletal structure. A discrete structural optimization problem can be formulated in the following form:

$$\begin{aligned} &\min && F(s) \\ &\text{subject to} && g_j(s) \leq 0 \quad j = 1,...,m \\ & && s_i \in R^d, \quad i = 1,...,n \end{aligned} \tag{1}$$

where F(s) and g(s) denote the objective and constraints functions respectively. R^d is a given set of discrete values, the design variables s_i (i=1,2,...,n) can take values only from this set. Skeletal structures are very common in engineering practice in order to cover long and/or wide span and column-free spaces such as stadiums, exhibition halls, airplane hangars, etc. Skeletal structures usually have the topology of single or multi-layered flat or curved grids that can be easily constructed. Most frequently the objective function is the weight or the volume of material of the structure and the constraints are the member stresses, nodal displacements, or frequencies.

2.1 Parametric optimization

In parametric optimization problems the aim is usually to minimize the weight of the structure under certain behavioural constraints on stresses and displacements. The design variables are most frequently chosen to be dimensions of the cross-sectional areas of the members of the structure. The parametric optimization methodology proceeds with the following steps: (i) At the outset of the optimization procedure the geometry, the boundaries and the loads of the structure under investigation have to be defined. (ii) The design variables, which may or may not be independent to each other, are also properly selected. Furthermore, the constraints are also defined at this stage in order to formulate the optimization problem as in eq. (1). (iii) A finite element analysis is then carried out and the displacements and stresses are evaluated. (iv) If a gradient-based optimizer, like the successive quadratic programming algorithm, is used then the sensitivities of the constraints and the objective function to small changes of the design variables are computed either with the finite difference, or with the semi-analytical method [3]. (v) The design variables are being optimized. If the convergence criteria for the optimization algorithm are satisfied, then the optimum solution has been found and the process is terminated, else the optimizer updates the design variable values and the whole process is repeated from step (iii).

2.2 Shape-topology-parametric optimization

Parametric optimization can be combined with shape and topology optimization. When shape optimization is considered the structural domain is not fixed but it has a predefined topology. In both cases of parametric and shape optimization a non-optimal starting topology can lead to sub-optimal results. To overcome this deficiency structural topology optimization needs to be taken into consideration, which allows the designer to optimize the layout or the topology of a structure by detecting and removing the low-stressed material in the structure which is not used effectively. The problem of shape-topology-parametric optimization of multi-layered skeletal structures is implemented in this work in three phases with the following sequence: shape, topology and parametric.

3 EVOLUTIONARY COMPUTATION

EC encompasses methods of simulating evolution processes on computing systems. Evolutionary algorithms belong to EC and represent the probabilistic category of optimization methods. The first attempt in the field of evolutionary computation took place in the sixties by a team of biologists and was focused in building a computer program that would simulate the process of evolution in nature. Evolutionary algorithms have been found capable to produce very powerful and robust search mechanisms although the similarity between these algorithms and the natural evolution is based on crude imitation of biological reality. The resulting evolutionary algorithms are based on a population of individuals, which are subjected to processes of mutation, recombination/crossover and selection.

GA and ES are the evolutionary algorithms employed in this study. Although, both methods represent two independently developed branches of EC and they are known since more than thirty years, the method of GA has gained much more interest than ES particularly during the past ten years. EC imitate biological evolution in nature and have three characteristics that differ from mathematical programming optimization methods: (i) In place of the usual deterministic operators, they use randomized operators: mutation, selection and recombination/crossover. (ii) Instead of a single design point, they work simultaneously with a population of design points in the space of design variables. (iii) They can handle continuous, discrete or mixed optimization problems.

In structural optimization problems, where the objective function and the constraints are highly non-linear functions of the design variables, the computational effort spent in gradient calculations required by the mathematical programming algorithms is usually large. In two studies by Papadrakakis et al. [3,9] it was found that probabilistic search algorithms are computationally efficient even if greater number of optimization cycles is needed to reach the optimum. These cycles are computationally less expensive than in the case of mathematical programming algorithms since they do not need gradient information. Furthermore, probabilistic methodologies were found, due to their random search, to be more robust in finding the global optimum, whereas mathematical programming algorithms may be trapped in local optima.

4 HP-OCP

The solution of real-world structural optimization problems can only be achieved with a synergy of the following actions during the numerical simulation and design procedure: (i) Using accurate and computationally-efficient models for the numerical modelling of the physical problem; (ii) Applying reliable and efficient metaheuristic optimization algorithms for improving the design procedure; (iii) Rational modelling of the system uncertainties (in the case of probabilistic formulation of the design problem) and (iv) Exploiting recent high performance computing (HPC) technology of workstations.

4.1 Graphical User Interface and Operating System

HP-OCP is based on object-oriented general-purpose code written in C# and specifically developed for the optimum design of civil engineering structures while it is the official simulation platform of the HP-OCP Intelligence Inc., CSI Engineering and the Institute of Structural Analysis and Seismic Research of the National Technical University of Athens. HP-OCP comes with a graphical user interface (GUI). GUI is highly configurable and allows tailoring to specific applications. It is also designed to communicate with third party software via an open application programming interface (OAPI) which must be capable to receive sets of input parameters from the GUI, determine the associated response, assess and return it to the optimization component. HP-OCP is easy to use and easy to integrate into any system. Combining a familiar Windows-based interface with a simple menu-driven system, HP-OCP integrates quickly into most existing environments. HP-OCP is compatible with the most popular operating systems, including Windows 95/98/ME/NT/2000/XP/7/8/10. Furthermore, HP-OCP can also be installed on UNIX platforms.

4.2 High Performance Computing Component

The use of MOA in structural optimization requires a number of FE structural analyses for the evaluation of the objective and constraint functions at each optimization step. An important characteristic of MOA that differs from other conventional optimization algorithms is that in place of a single design point the MOA work simultaneously with a population of design points in the space of variables. This allows for a straightforward implementation of the optimization procedure in parallel computer environments (natural parallelization). Since a number of FE analyses of the structure can be performed independently and concurrently, a complete finite element analysis can be allocated to a processor without the need for inter-processor communication during the solution phase. Therefore, the parallelization of the metaheuristics is based on the fundamental premise that each individual in the population of the offsprings represents an independent unit of all design variables and therefore its function evaluation can be done independently and concurrently.

Extensive research in the last decades led to the development of efficient iterative and direct sparse solution methods for linear finite element simulations in structural mechanics. A number of computer hardware manufacturers provide parallel sparse solvers in the scientific software libraries for their computers. Sparse solvers demonstrate high computational efficiency and achieve good parallel performance in few processor configurations of shared memory multiprocessors. However, they do not scale well in computer architectures with many processors. Furthermore, excessive inter-processor communication requirements constitute a serious drawback for their efficient implementation in distributed memory environments. Iterative parallel methods on the other hand entail low memory and communication requirements, but frequently raise the issue of robustness and computational efficiency. Research in the field of iterative high performance methods has been significantly boosted by domain decomposition methods (DDM). In the last decade, several, high performance iterative DDM have been proved very efficient in Computational Structural Mechanics [10].

4.3 Implementation of the domain decomposition method

In this paper we employ the one-level domain decomposition method, with fully redundant Lagrange multipliers. The optimal configuration of this method for the skeletal structures considered in this paper employs the lumped preconditioner and superlumped stiffness scaling in both the projector and the preconditioner [11]. All computational kernels of the one-level domain decomposition method are performed in parallel, except the factorization and forward and backward substitutions required by the projection steps. Details concerning such an implementation can be found in [12].

The subdomain semi-definite problems are handled by a geometric-algebraic method. Furthermore, the present solver is equipped with a fast preprocessing step for reducing the number of mechanisms in subdomains of the skeletal structures. First, all member elements that lie on the interface between subdomains are detected. These elements are then split to as many elements as the number of neighboring subdomains. The new generated elements have the Young modulus of the parent element divided by the number of common subdomains and the same cross section as the parent element. The new elements are then assigned to the neighboring subdomains and substitute the parent element. This preprocessing step regularizes interface stiffness coefficients and ensures a small number of zero energy modes in each subdomain, thus reducing the CPU time of the projection steps. The adopted one-level domain decomposition implementation is proven very efficient in solving large-scale pin-jointed skeletal structures and is effectively combined with parallel evolutionary algorithms for optimization problems.

4.4 Computer implementation of MOA

The use of MOA in structural optimization requires a number of FE structural analyses for the evaluation of the objective and constraint functions at each optimization step. An important characteristic of MOA that differs from other conventional optimization algorithms is that in place of a single design point the MOA work simultaneously with a population of design points in the space of variables. This allows for a straightforward implementation of the evolution procedure in parallel computer environments (natural parallelization). Since a number of FE analyses of the structure can be performed independently and concurrently, a complete finite element analysis can be allocated to a processor without the need for inter-processor communication during the solution phase. Therefore, the parallelization of the MOA is based on the fundamental premise that each individual in the population of the offsprings represents an independent unit of all design variables and therefore its function evaluation can be done independently and concurrently.

In a distributed memory computing environment this implementation can be realized provided that there is enough memory at each processor to accommodate the storage required by the finite element simulation. There is also need for a host processor to accumulate all information from the other p-1 processors in order to select the parents of the next generation. In a shared-memory environment, however, there is no need for a host processor, while no storage limitations are imposed by each processor but on the total memory of the computer which affects the number of processors that can be activated. The parallel strategies proposed in the current study, are properly modified to address the characteristic features of MOA. In MOA the natural parallelization is straight forward apart from a small load imbalance that may occur when the size of the problem assigned to the processors varies, as may happen in topology optimization.

5 NUMERICAL TESTS

Two numerical tests were performed to observe the time reduction achieved from paralleling MOA. A concrete building was chosen for the first test case and a steel structure for the second. Finite element analysis and design was performed in ETABS for both structures while the rest of the optimization was handled by HP-OCP.

5.1 Test Case 1: Three story concrete building

The first case is the three story concrete building depicted in Figure 1, that consists of 378 frame elements, 93 area elements and a total of 15 different section properties for both types. The building overall dimensions were 32.6 m wide with an 8.75 m width with three different concrete types of C20/25, C25/30 and C30/37. The loading types assigned were self-weight, super dead, live load and earthquake along the x and y directions and the design codes Eurocode

EC2 and EC8.

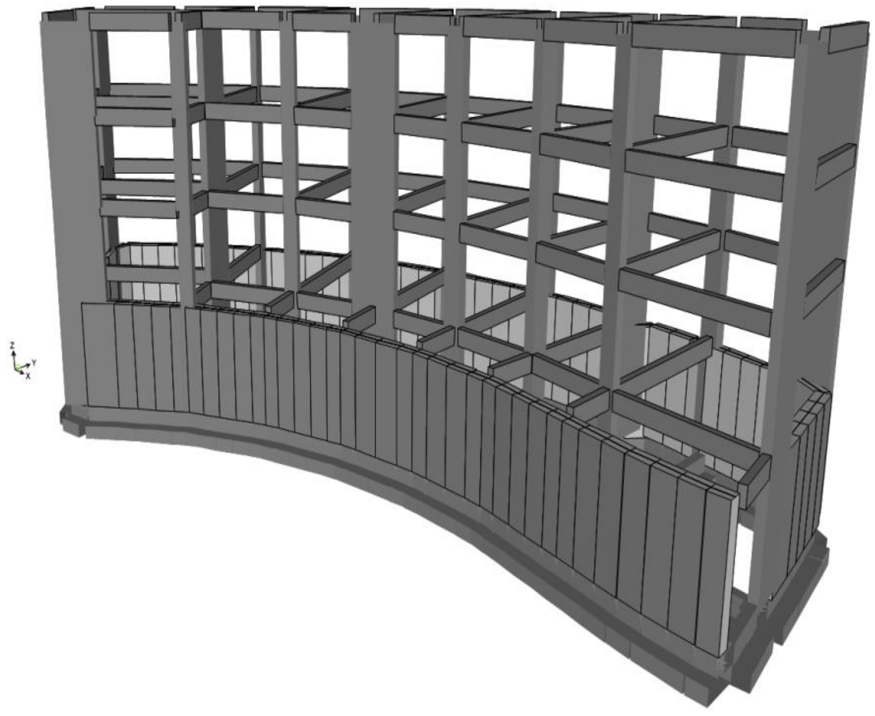

Figure 1: test case 1 concrete structure

The total material cost was set as the Objective function of the optimization procedure and as design variables all the different dimensions of the section properties. The total number of iterations performed during the optimization procedure were 300 reducing the objective function from 46,500 Euros to 34,060 Euros. The total serial optimization time was 106 minutes. By paralleling the optimization procedure and utilizing four concurrent optimization procedures the time needed dropped to 87 minutes achieving reduction of about 18%. As it can be seen the reduction was not proportional to the number of separate procedures utilized.

5.2 Test Case 2: Steel structure

The second case tested is a steel structure as depicted in Figure 2, that consists of 3,294 steel frame elements with 9 different section properties. The structure's opening is equal to 12.35 m and 19.94 m width with one steel type S355. The loading types assigned were self-weight, live load and multiple different wind loads and the design codes Eurocode 3.

Similar to test case 1, the objective function of the optimization procedure was set as the total structure's material cost and the design variables all the different section properties dimensions. The total number of iterations performed during the optimization procedure were again 300 reducing the objective function from 10,180 Euros to 9,530 Euro. The total serial optimization time was 252 minutes. By paralleling the optimization procedure and utilizing four concurrent optimization procedures the time needed dropped to 194 minutes achieving reduction of about 23%. As we can see again the reduction was not proportional to the number of separate procedures utilized.

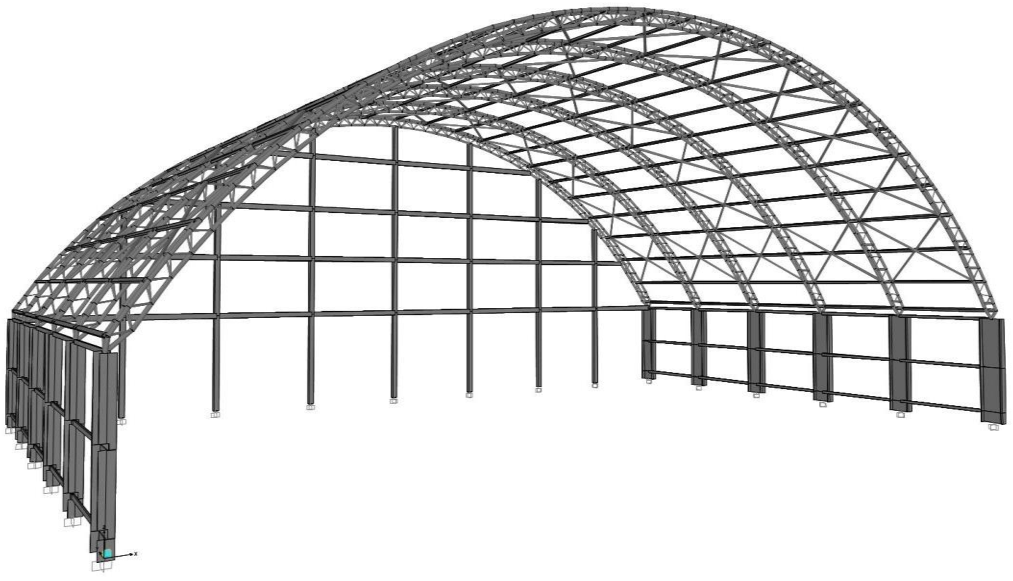

Figure 2: test case 2 steel structure

6 CONCLUSIONS

An important characteristic of MOAs that makes them differ from other conventional optimization algorithms is that in place of a single design point they work simultaneously with a population of design points in the space of design variables. This allows for a straightforward implementation of the evolution procedures in parallel computer environments. Since a number of finite element analyses of the structure can be performed independently and concurrently, a complete finite element analysis can be assigned to a processor without the need for inter-processor communication during the solution phase. It has been shown that the computational efficiency of MOAs examined in this study may be further enhanced substantially with the implementation of parallel domain decomposition algorithms for the solution of each of the repeated finite element problems encountered at each optimization step of the evolutionary algorithm procedure.

7 ACKNOWLEDGMENT

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T1EDK-05603-"HP-OCP" - Holistic, High Performance Optimization Computing Platform»).

8 REFERENCES

[1] OptiStructure Ltd. http://optistructure.com/, (January 2021).
[2] N.D. Lagaros, A general purpose real-world structural design optimization computing platform, *Structural and Multidisciplinary Optimization*, 49: 1047–1066, (2014).
[3] M. Papadrakakis, Y. Tsompanakis, N.D. Lagaros. Structural shape optimization using Evolution Strategies. *Engineering Optimization*, 31: 515-540, (1999).

[4] M. Papadrakakis, N.D. Lagaros, Y. Fragakis, Parallel computational strategies for structural optimization, *International Journal for Numerical Methods in Engineering*, 58(9): 1347-1380, (2003).

[5] N.D. Lagaros, An efficient dynamic load balancing algorithm, *Computational Mechanics*, 53(1): 59-76, (2014).

[6] H. Adeli and S. Kumar. Concurrent structural optimization on massively parallel supercomputer. *Journal of Structural Engineering*, 121(11), 1588-1597, (1995).

[7] C. Farhat and F.-X. Roux. A method of finite element and interconnecting and its parallel solution algorithm. Int. J. Numer. Meth. Engng., 32, 1205-1227, (1991).

[8] C. Farhat and F.-X. Roux. Implicit parallel processing in structural mechanics. *Comput. Mech. Adv.*, 2, 1-124, (1994).

[9] M. Papadrakakis, N.D. Lagaros and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. *Comp. Meth. Appl. Mechanics & Engrg*, 156, 309-333, (1998).

[10] P. LeTallec. Domain-decomposition methods in computational mechanics. *Comput. Mech. Adv.*, 1, 121-220, (1994).

[11] M. Bhardwaj, D. Day, C. Farhat, M. Lesoinne, K. Pierson and D. Rixen. Application of the FETI Method to ASCI Problems: Scalability results on one-thousand processors and discussion of highly heterogeneous problems. *International Journal for Numerical Methods in Engineering*, 47, 513-536, (2000).

[12] C. Farhat, K. Pierson and M. Lesoinne. The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Comput. Methods Appl. Mech. Engrg.*, 184, 333-374, (2000).