SEECCM III
3rd South-East European Conference on Computational Mechanics-
*an ECCOMAS and IACM Special Interest Conference*
M. Papadrakakis, M. Kojic, I. Tuncer (eds.)
Kos Island, Greece, 12–14 June 2013

# ROBUST DESIGN OPTIMIZATION IN PARALLEL COMPUTING ENVIRONMENT

**Nikos D. Lagaros and Manolis Papadrakakis**

Institute of Structural Analysis & Antiseismic Research
School of Civil Engineering
National Technical University of Athens
Zografou Campus, Athens 157 80, Greece
e-mail: {nlagaros,mpapadra}@central.ntua.gr

**Keywords:** Parallel computing, dynamic load balancing algorithm, exploitation of the computing strategies, structural robust design optimization, multi-objective optimization, metaheuristics.

**Abstract.** *In engineering problems, randomness and uncertainties are inherent. Robust design procedures, formulated in the framework of multi-objective optimization, have been proposed in order to take into account the various sources of randomness and uncertainty. These design procedures require orders of magnitude more computational effort than the conventional analysis or the optimum design processes since a very large number of finite element analyses is required to be dealt. It is therefore an imperative need to exploit the capabilities of the computing resources in order to deal with this kind of problems. In particular, parallel computing is implemented at the level of the metaheuristic optimization, by exploiting the physical parallelization feature of the nondominated sorting evolution strategies method, as well as at the level of the repeated structural analyses required for assessing the behavioural constraints and for calculating the objective functions. In this study an efficient dynamic load balancing algorithm (DLBA) for the optimum exploitation of the available computing resources is proposed and, without loss of generality, DLBA is applied to the task of computing the desired feasible Pareto front. In such problems the computation of the complete Pareto front with feasible designs only, constitutes a very challenging task. The proposed algorithm achieves linear speedup factors and almost 100% speedup factor values with reference to the sequential procedure.*

## 1 INTRODUCTION

Real world engineering systems always have imperfections or deviations from the nominal state which affect their response. These uncertainties are reflected in any optimized design obtained via an optimization methodology and thus it can never be materialized in an absolute way. A deterministic formulation of a structural optimization problem ignores the unavoidable scatter of the structural parameters; hence, the performance of the optimum designs obtained through deterministic formulations may be much different than expected. In recent years, design optimization studies have become increasingly concerned with the treatment of uncertainty and randomness and various stochastic formulations for the optimum design of structures have been proposed [1]. The methodological context for defining such designs is referred either as robust design optimization (RDO) or as reliability-based optimization (RBO). The aim of RDO formulations [2-5] is to improve product quality or stabilize performances by minimizing the effects of variations. The main goal of RBO formulations is to design for safety with respect to extreme events by determining optimum designs satisfying both deterministic and probabilistic constraints. In the combined reliability-robust design optimization (RRDO) formulation proposed in [6], which is an extension of the other two formulations, the probability of failure of the structure and/or the probability of violation of the constraints are taken into account as a set of probabilistic constraints together with the deterministic constraints considered by the RDO formulations, both RDO and RRDO are formulated as two-objective optimization problems. This type of problems is extremely computationally intensive since a vast number of finite element (FE) analyses should be performed.

In the present work, the nondominated sorting evolution strategies II (NSES-II) [7] method is used for the solution of the two-objective RDO sizing optimization problems of large-scale three-dimensional skeletal structures, while the required random characteristics of the structural response are computed with the Monte Carlo simulation method using the Latin hypercube sampling technique. It is therefore appropriate to perform the required computations in a parallel or distributed computing environment in order to be able to conduct RDO studies efficiently. In particular, the computations required by the NSES-II method are implemented in a two level parallelization scheme: (i) at the level of the optimization procedure exploiting the physical parallelization feature of the NSES-II method and (ii) at the level of the solution of every finite element analysis required for calculating the constraint and objective functions in order to assess each candidate design vector. The solution of the structural analysis problems encountered is performed with the finite element tearing and interconnecting (FETI) domain decomposition method [8-10]. The two objective functions considered in the RDO formulation are the material volume of the structure and the variance of a structural response parameter, while uncertainty on loads, material properties and member cross-sections is taken into consideration. Furthermore, each candidate design is checked whether it satisfies the corresponding code requirements for the design of structures [11]. In such problems it is desirable covering the Pareto front with feasible designs only, this task constitutes a very challenging task. In an effort to achieve the desired feasible Pareto front it is proposed an efficient dynamic load balancing algorithm for the optimum exploitation of the available computing resources. The proposed algorithm achieves linear speedup factors and almost 100% speedup factor values with reference to the sequential procedure. In order to prove the efficiency of the proposed dynamic load balancing algorithm synchronous and asynchronous variants of NSES-II method are implemented; while three constraint handling techniques are also considered.

It should be stated that the implementation of the proposed dynamic load balancing algorithm is not limited to the solution of RRDO problems only. It is applicable in any problem where an extremely large number of FE analyses are to be dealt. For example: (i) Structural reliability analysis problems implementing simulation based procedures into a heterogeneous computing environment (i.e. heterogeneous cloud computing platform). (ii) Metaheuristic structural design optimization carried out into a heterogeneous computing environment. (iii) Stochastic structural design optimization where the number of simulations required for assessing a design is not fixed; it depends on how small probabilities are sought. It should be stated also that the proposed method is applicable both in the case of the physical parallelization and in the case of a two level parallelization scheme where domain decomposition FE solution methods are implemented in conjunction with the physical parallelization characteristic of the problem at hand**.**

## 2 ROBUST DESIGN OPTIMIZATION

A robust structural design optimization problem is formulated as a two-objective optimization problem where in addition to the objective function representing the initial construction cost or the structural weight, the influence of the random nature of structural parameters and loading conditions on the structural performance is also considered as an objective function [12].

### 2.1 Formulation of the optimization problem

The mathematical description of the generic RDO problem implemented in this study is formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{s}\in\mathcal{F}} && [W(\boldsymbol{s}),\ \sigma_u(\boldsymbol{s},\boldsymbol{r})]^{\mathrm{T}} \\ &\text{subject to} && \bar{g}_j(\boldsymbol{s},\boldsymbol{r}) \le 0, \quad j = 1,\dots,k \\ &\text{where} && \boldsymbol{r} \sim \mathrm{N}(\boldsymbol{\mu}_{\mathrm{r}},\boldsymbol{\sigma}_{\mathrm{r}}^2) \end{aligned} \tag{1}$$

where $W(\boldsymbol{s})$ and $\sigma_{\mathrm{u}}(\boldsymbol{s},\boldsymbol{r})$ are the two objectives to be minimized, corresponding to the weight of the structure and the standard deviation of the structural response, respectively; $\boldsymbol{s}$ and $\boldsymbol{r}$ are the vectors of the design and random variables; $\bar{g}_j(\boldsymbol{s},\boldsymbol{r})$ are the mean values of the $k$ constraint functions ($j$=1,2,…,$k$); $\mathcal{F}$ is the feasible region, defined as the region of the design space for which the constraints of Eq. (1) are satisfied:

$$\mathcal{F}=\{\boldsymbol{s}\in R^d \mid \bar{g}_j(\boldsymbol{s},\boldsymbol{r}) \le 0 \quad \mathrm{j}=1,\dots,\mathrm{k}\} \tag{2}$$

while $R^d$ is a discrete set from which the design variables $\boldsymbol{s}$ take values.

### 2.2 Solving the multi-objective optimization problem

Several methods have been proposed for treating structural multi-objective optimization problems [13-15]. For the present study an improved version of the genetic algorithm based NSGA-II method [16] is implemented where the genetic algorithms are replaced by the mixed evolution strategies (ES) algorithm. The ES algorithm has been proven very efficient for solving single objective structural optimization problems [17]. The resulting algorithm, proposed in [7], is denoted as NSES-II($\mu+\lambda$) or NSES-II($\mu,\lambda$) depending on the selection scheme.

In a mixed optimization problem, with discrete and continuous design variables, solved with the ES algorithm, each individual is equipped with a set of parameters:

$$\begin{aligned} &\mathbf{a} = [(\mathbf{s}_{\mathrm{d}},\boldsymbol{\gamma}),(\mathbf{s}_{\mathrm{c}},\boldsymbol{\sigma},\boldsymbol{\alpha})] \in (\mathrm{I}_{\mathrm{d}},\mathrm{I}_{\mathrm{c}}) \\ &\mathrm{I}_{\mathrm{d}} = \mathrm{D}^{\mathrm{n_d}} \times \mathrm{R}_+^{\mathrm{n}_\gamma} \\ &\mathrm{I}_{\mathrm{c}} = \mathrm{R}^{\mathrm{n_c}} \times \mathrm{R}_+^{\mathrm{n}_\sigma} \times [-\pi,\pi]^{\mathrm{n}_\alpha} \end{aligned} \tag{3}$$

where $\mathbf{s}_{\mathrm{d}}$ and $\mathbf{s}_{\mathrm{c}}$ are the vectors of discrete and continuous design variables, respectively. Vectors $\boldsymbol{\gamma}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\alpha}$ are the distribution parameter vectors. Vector $\boldsymbol{\gamma}$ corresponds to the variances of the Poisson distribution, vector $\boldsymbol{\sigma} \in \mathrm{R}_+^{\mathrm{n}_\sigma}$ corresponds to the standard deviations ($1 \le \mathrm{n}_\sigma \le \mathrm{n}_{\mathrm{c}}$) of the normal distribution while vector $\boldsymbol{\alpha} \in [-\pi,\pi]^{\mathrm{n}_\alpha}$ corresponds to the inclination angles ($\mathrm{n}_\alpha = (\mathrm{n}_{\mathrm{c}}-\mathrm{n}_\sigma/2)(\mathrm{n}_\sigma-1)$) that define a linear correlation of mutations of the continuous design variables $\mathbf{s}_{\mathrm{c}}$.

Let $\mathbf{B}_p^{(g)} = \{\mathbf{a}_1,\dots,\mathbf{a}_\mu\}$ denotes a parent population of individuals at the gth generation. The genetic operators used in the mixed discrete ES method are denoted by the following mappings:

$$\begin{aligned} &\text{rec} \;:\; (\mathrm{I_c})^{\mu}_{(g)} \rightarrow (\mathrm{I_c})^{\lambda} \text{ recombination} \\ &\text{mut} \;:\; (\mathrm{I_c})^{\lambda} \rightarrow (\mathrm{I_c})^{\lambda} \text{ mutation} \\ &\text{sel}^{\mathrm{k}}_{\mu} \;:\; (\mathrm{I_c})^{\mathrm{k}} \rightarrow (\mathrm{I_c})^{\mu}_{(g+1)} \text{ selection, } \mathrm{k} \in \{\lambda,\ \mu+\lambda\} \end{aligned} \tag{4}$$

A single iteration of the mixed ES, which is a step from the parent population $\mathbf{B}_p^g$ to the next generation, $\mathbf{B}_p^{g+1}$ is modelled by the mapping:

$$\text{opt}_{\mathrm{ES}} \;:\; (\mathrm{I_c})^{\mu}_{(g)} \rightarrow (\mathrm{I_c})^{\mu}_{(g+1)} \tag{5}$$

NSES-II(μ+λ) is a generational algorithm, where a parent population is used in order to create the offspring population; then, both populations are combined to define the new parent population. An alternative to the generational metaheuristic (evolution strategies in the present study) is the steady-state (SS) evolution strategies, where there is only a single population (not a parent and an offspring one). The new individuals are incorporated directly in the evolution process; thus, parents and offspring coexist in the same population. The steady-state evolution strategies algorithm is characterized by incrementing the intensification capability of the search process. A steady-state variant of NSES-II(μ+λ) can easily be implemented by using an offspring population of size one. This means that the ranking and crowding procedures are applied each time a new individual is created, thus the computational complexity of the algorithm increases. In the rest of this work the steady-state variant is referred as NSES-II$_{\mathrm{ss}}$ while the original algorithm is referred as NSES-II$_{\mathrm{gen}}$.

## 2.3 Parallel metaheuristic multi-objective optimization

Multi-objective optimization problems require searching for a whole set of optimum solutions (Pareto front) instead of a single optimum solution and therefore the need for parallel computing in metaheuristic multi-objective (MMO) optimization is even greater. A direct application of a MMO method (without loss of the generality the NSES-II(μ+λ) algorithm considered in the current study) in parallel computing environment is to exploit the physical parallelization feature of the MMO method into a master-slave scheme. In this approach, the master processor performs the steps of the NSES-II(μ+λ) algorithm while both master and workers processors evaluate the objective functions of the new individuals assigned. This parallel variant is referred as synchronous generational NSES-II(μ+λ) and it is denoted as $\text{NSES-II}^{syn}_{gen}$. According to this scheme, *λ* individuals are created and sent to the master and workers processors for assessment; when all of them have been assessed and returned to the master processor the new generation is completed. The performance of $\text{NSES-II}^{syn}_{gen}$ algorithm depends on the number of available processors while it does not take advantage of the number of processors if it is higher than the population size *λ*. The asynchronous variant takes advantage of the number of the processors that are higher than *λ*. The idea in the asynchronous approach is to use all the available processors, which use to be larger than the population size of the algorithm. The asynchronous generational NSES-II(μ+λ), denoted as $\text{NSES-II}^{asy}_{gen}$, is based on the synchronous variant but $N_{proc}$ individuals are created instead of *λ*, where $N_{proc}$ is the available number of processors. Furthermore, when an assessed individual is received by the master, a new one is generated and sent for assessment to an idle processor. The $\text{NSES-II}^{asy}_{gen}$ algorithm is still generational, because when the offspring population is filled, a new generation is completed. In $\text{NSES-II}^{asy}_{gen}$ the master does not have to wait until all the individuals of a generation have been assessed. As a consequence, the search capabilities of this algorithm are different compared to $\text{NSES-II}^{syn}_{gen}$, since it is possible that individuals generated later to be inserted into the evolution process before than individuals generated earlier. As it was mentioned before, the steady-state scheme in metaheuristics allows improving the exploitation of the search, so the purposes of the asynchronous steady-state NSES-II denoted as $\text{NSES-II}^{asy}_{SS}$ is twofold: to achieve this behaviour in a master-slave NSES-II and to reduce the time needed to solve MOPs using as many processors as possible. In $\text{NSES-II}^{asy}_{SS}$, the master processor also generates as many individuals as available processors. Once each evaluated solution is received, it is inserted in the population applying ranking and crowding. As in $\text{NSES-II}^{asy}_{gen}$, when idle workers are detected, new individuals are created and sent for evaluation to an idle processor.

Lee and Hajela [18] described an adaptation of genetic algorithms for multidisciplinary optimization problems in parallel computing environment. Coello Coello *et al.* [19] discussed different paradigms for parallel implementation of MMO methods, while new concepts on migration, replacement and niching schemes were discussed by VanVeldhuizen *et al.* [20]. de Toro Negro *et al.* [21] proposed a parallel genetic algorithm with a structured population in the form of a set of islands, while Wilson and Moore [22] proposed a parallel algorithm using multiple optimization criteria, independent populations of individuals which consist of multiple chromosomes. Durillo *et al.* [23] studied three parallel approaches (a synchronous and two asynchronous) for the nondominated sorting genetic algorithms II (NSGA-II) method based on the master-worker paradigm. Kipouros *et al.* [24] presented a multi-objective variant of the tabu search optimization algorithm implemented in parallel computers for the aerodynamic design optimization of turbo machinery blades. Bharti *et al.* [25] focused on the optimal design of morphing aircraft wings employing a wing structure composed of an internal layout of cables and struts using a parallelized variant of the NSGA-II method. Fan and Chanh [26] tested the efficiency of a parallel particle swarm multi-objective evolutionary algorithm in a variety of test functions taken from the literature. Nebro and Durillo [27] proposed a thread-based parallel version of a multi-objective evolutionary algorithm, while Zhou and Tan [28] applied a parallel version of multi-objective particle swarm optimization on graphics processing units. Mezmaz *et al.* [29] investigated the problem of scheduling precedence-constrained parallel applications on heterogeneous computing systems such as cloud computing infrastructures and proposed a parallel bi-objective hybrid genetic algorithm.

The above mentioned implementations are limited to the exploitation of the physical parallelization feature of the MMO methods. Furthermore, the feasibility of the designs comprising the Pareto front curve was satisfied with the use of constraint handling procedures that does not guaranty 100% feasibility. In this study the parallel implementation is performed both at the level of MMO method as well as at the level of the repeated FE analyses required for calculating the objective and constraint functions. This was performed implementing both the synchronous and asynchronous schemes with the generational or the steady-state implementation while a domain decomposition FE solution method was also considered. In particular, both $\mathrm{NSES\text{-}II}_{gen}^{syn}$ and $\mathrm{NSES\text{-}II}_{SS}^{syn}$ algorithms are implemented in this study. Furthermore, the feasibility of the designs is satisfied by implementing three constraint handling techniques either by rejecting any infeasible design in order to produce reliable Pareto front curves or a superiority of feasible points technique.

### 2.4 Constraint Handling Techniques

Metaheuristics are initially developed to solve unconstrained optimization problems; however, during the last decades several methods have been proposed for handling constrained optimization problems as well. A detailed survey can be found in the works by Coello Coello [30,31]. Indicatively three methods have been selected in this study in order to prove that the efficiency of the proposed dynamic load balancing algorithm is not dependant on the constraint handling technique. In particular in this study methods belonging to the methods of superiority of feasible points and rejection of infeasible points have been implemented.

## 3 OPTIMUM EXPLOITATION OF PARALLEL COMPUTING STRATEGIES

The use of metaheuristics in RDO problems requires a large number of FE analyses for the calculation of the objective and constraint functions at each optimization step. This is due to the characteristic of metaheuristic multi-objective optimization algorithms that they work simultaneously with a population of design points in the space of the design variables. This allows for a straightforward implementation of the optimization procedure in parallel computing environments (physical parallelization). Therefore, in the case of the NSES-II($\mu$+$\lambda$) algorithm adopted for solving the RDO problem, $\mathrm{N}_{\mathrm{anal}} = \lambda \cdot \mathrm{N}_{\mathrm{sim}}$ FE analyses can be performed independently and concurrently in every generation, where $\lambda$ is the size of the population and $N_{sim}$ is the number of the LHS samples considered. Thus, a complete finite element analysis can be allocated to a processor, without the need for inter-processor communication during the solution phase, and the analyses can be performed simultaneously on the available processors.

## 3.1 Parallel computing strategies

In multi-objective optimization there exists no unique design which represents the optimum for all criteria. Thus, the common optimality condition used in single-objective optimization is replaced by the so called Pareto optimum. A design vector $\underline{\boldsymbol{s}} \in \mathcal{F}$ is called Pareto optimum for a multi-objective optimization problem with *m* objective functions (all to be minimised) if there is no other design vector $\boldsymbol{s} \in \mathcal{F}$ such that:

$$\begin{aligned} & f_i(\boldsymbol{s}) \leq f_i(\underline{\boldsymbol{s}}) \text{ for } i = 1,\dots,m \\ \text{with } & f_i(\boldsymbol{s}) < f_i(\underline{\boldsymbol{s}}) \text{ for at least one objective } i \end{aligned} \tag{6}$$

The geometric locus of the Pareto optimum solutions is called Pareto front and represents the solution of the optimization problem with multiple objectives. A typical Pareto front curve is depicted in Figure 1 for two objective functions $f_1(x)$ and $f_2(x)$ both to be minimized. It can be seen that the definition of Eq. (6) is satisfied for designs *a* and *c*, since there exists no feasible design that is better than the two designs with reference to both objective functions. On the other hand the feasible designs of the hatched region are better compared to the design *b* with reference to both objective functions. Consequently, designs *a* and *c* belong to the Pareto front while design *b* does not constitute a Pareto optimum. The checks described in Eq. (6) can be performed only if the designs **s** and $\underline{\boldsymbol{s}}$ are feasible. Therefore in MMO only feasible solutions are accepted, hence the following three cases can occur at each generation of a MMO procedure: (i) All members of the population are feasible; (ii) all members of the population are infeasible and (iii) some of the members are feasible and some are not feasible.

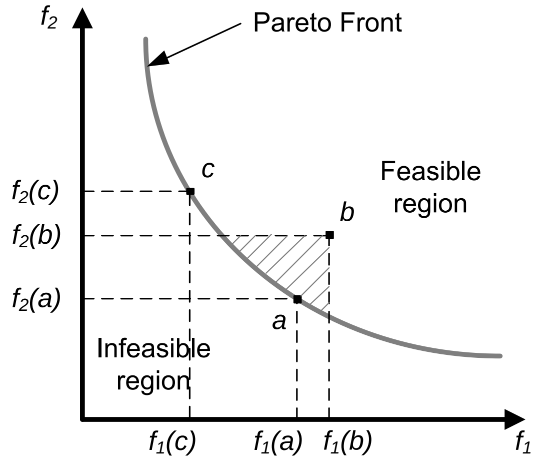


Figure 1: Typical Pareto front curve

According to the NSES-II(μ+λ) optimization procedure a population of λ vectors is produced in every generation. In order to check their feasibility in the framework of a robust design optimization, $N_{anal} = \lambda \cdot N_{sim}$ FE analyses are performed. There are two different schemes to perform these analyses in parallel within a generation step:

i) Perform the optimization steps in serial mode and the corresponding FE analysis in parallel mode on the $N_{proc}$ processors with the FETI solver.
ii) Perform in parallel mode both the optimization steps and the corresponding FE analyses. In this case the $N_{anal} = \lambda \cdot N_{sim}$ FE analyses, required in every generation of the optimization procedure, can be performed concurrently and each solution of the finite element equations can also be performed in parallel mode with the FETI solver. These analyses correspond to the offspring vectors which are generated during the evolution process and according to the constraints check they are feasible or infeasible. In the case of an infeasi-

ble offspring vector a new one is being generated until a feasible one is found. The feasibility check requires new $N_{sim}$ FE analyses for each new offspring vector. Therefore, if $\lambda_i$ out of $\lambda$ designs were found infeasible, then $N_{anal} = \lambda_i \cdot N_{sim}$ new FE analyses must be carried out at this phase of the generation step. The optimal load balancing of this task is not a straightforward procedure. It is clear that, during the evolution procedure for reaching the optimum, different number of FE analyses that can be performed independently and concurrently are required while it is not obvious how many processors will be involved for the solution of each finite element problem to perform the constraints check. For this reason it is necessary to propose a dynamic load balancing algorithm (DLBA) for the fully exploitation of the available computing resources at any phase of the offspring generation of the optimization procedure.

## 3.2 Dynamic load balancing algorithm

In this section an optimization problem is formulated in order to achieve the optimum dynamic load balancing at each phase during the generation step of the optimization procedure. This problem is stated as follows: define the optimum way to accommodate $N_{anal}$ FE analyses, which can be performed independently and/or concurrently, into a parallel computing environment composed by $N_{proc}$ processors with the least required computing time. In the case of a cluster composed by a multi-core node environment, $N_{proc}$ = number of nodes × number of cores. According to the proposed DLBA every FE analysis can be alternatively performed on $N_{sub}$ different domain decompositions varying on the number of subdomains. It is assumed, without this being restricting, that each subdomain is handled by one processor. Therefore, the dynamic load balancing optimization procedure is formulated as to define the optimum number of FE analyses performed implementing the ith (i=1,…,$N_{sub}$) decomposition, that requires the least computing time to carry out $N_{anal}$ FE analyses.

The full exploitation of the available resources can be sated as an optimization problem as follows:

$$\begin{aligned} &\min_{\boldsymbol{n}_{anal}} && Time(N_{anal}) \\ & && \boldsymbol{n}_{anal} = \left[n_{anal}(1),\, n_{anal}(2),\, \ldots,\, n_{anal}(N_{sub})\right]^T \\ & && n_{anal}(\mathrm{i}) \in [0, \mathrm{N}_{anal}],\ i=1,\ldots,N_{sub} \\ &\text{and} && \sum_{i=1}^{N_{sub}} n_{anal}(\mathrm{i}) = \mathrm{N}_{anal} \end{aligned} \tag{7}$$

where $n_{anal}$(i) ($i$=1, 2, …, $N_{sub}$) are the design variables of the dynamic load balancing optimization problem and corresponds to the number of FE analyses performed implementing the ith decomposition. The total computing time (*Time*) is calculated according to a two-step procedure as follows:

*Step 1*

In the first step the time required for performing the FE analyses for every possible decomposition that can be accommodated by the entire cluster of processors ($N_{proc}$) without any remaining idle processors, is calculated. This time ($Time_1$) is obtained as follows:

$$n_{cluster}(i) = \mathrm{int}\left[\frac{n_{anal}(i) \cdot n_{sub}(i)}{N_{proc}}\right] \tag{8a}$$

$$Time_1 = \sum_{i=1}^{N_{sub}} n_{cluster}(i) \cdot t_{sub}(i) \tag{8b}$$

where $n_{cluster}$(i) denotes the number of times the cluster of processors will be used to perform FE analyses implementing $n_{sub}$(i) decomposition, $n_{sub}$(i) is the number of subdomains that the structure is subdivided (each subdomain is handled by one processors) according to the ith decomposition, $t_{sub}$(i) is the time required for performing a single analysis implementing FETI with $n_{sub}$(i) decomposition, while $Res_{anal}$(i):

$$Res_{anal}(i) = n_{anal}(i) - n_{cluster}(i) \cdot \frac{N_{proc}}{n_{sub}(i)} \tag{9}$$

denotes the residual FE analyses that remain to be performed implementing the $n_{sub}$(i) decomposition.

*Step 2-Syncronous Implementation*
In the second step, *Time*$_2$ defined the time required to perform the residual FE analyses that have not been performed in *Step 1*, in pursue of feasible solutions by the design vectors. This step is described in Figure 2, where in every use of the entire cluster of processors, groups of the residual FE analyses implementing different model partitioning types are accommodated by a part of the cluster. With the unused part of the cluster additional FE analyses are performed implementing other partitioning types ordered from the type with the least number of subdomains to that with the greater one.

```
Begin
    Repeat
        For $i := 1$ To $N_{sub}$ -1 Do Begin
            $n_{cluster}(\mathrm{i}) = \mathrm{int}\left[\frac{Res_{anal}(\mathrm{i}) \cdot \mathrm{n}_{sub}(i) + Res_{anal}(\mathrm{i}+1) \cdot \mathrm{n}_{sub}(i+1)}{N_{proc}}\right]$
            $Time_2 = Time_2 + n_{cluster}(i) \cdot t_{sub}(i)$
            $Res_{anal}(\mathrm{i}+1) = Res_{anal}(\mathrm{i}+1) - \frac{[n_{cluster}(i) \cdot N_{proc} - Res_{anal}(i) \cdot n_{sub}(i)]}{n_{sub}(i+1)}$
        End
    Until all analyses are performed
End
```

(a)

```
Begin
    Repeat
        For $i := 1$ To $N_{sub}$ -1 Do Begin
            $n_{cluster}(\mathrm{i}) = \mathrm{int}\left[\frac{Res_{anal}(\mathrm{i}) \cdot \mathrm{n}_{sub}(i) + Res_{anal}(\mathrm{i}+1) \cdot \mathrm{n}_{sub}(i+1)}{N_{proc}}\right]$
            $Time_2 = Time_2 + n_{cluster}(i) \cdot t_{sub}(i)$
            $Res_{anal}(\mathrm{i}+1) = Res_{anal}(\mathrm{i}+1) - \frac{[n_{cluster}(i) \cdot N_{proc} - Res_{anal}(i) \cdot n_{sub}(i)]}{n_{sub}(i+1)}$
        End
    Until all analyses are performed
End
```

(b)

Figure 2: The second step for the calculation of the required computing time for the case of (a) Synchronous and (b) Asynchronous Implementation

*Step 2-Asynchronous Implementation*
In the second step, *Time*$_2$ defined the time required to perform the residual FE analyses that have not been performed in *Step 1*, in pursue of feasible solutions by the design vectors. This step is described in Figure 2, where in every use of the entire cluster of processors, groups of the residual FE analyses implementing different model partitioning types are accommodated by a part of the cluster. With the unused part of the cluster additional FE analyses are performed implementing other partitioning types ordered from the type with the least number of subdomains to the that with the greater one.

## 4 NUMERICAL RESULTS

For the purposes of this study, the robust design of a double-layered space truss aircraft hangar is considered.

### 4.1 Description of the test example

The hangar is intended to service three planes at a time requiring a total clear width of 194 m and a clear depth of 89 m, therefore an area of 100×200 m$^2$ is designed to be covered. The space truss that consists of 51448 elements, 13031 nodes and 38961 degrees of freedom. Due to engineering practice demands, the members are divided into groups having the same structural properties. This grouping of elements results in a trade-off between the use of more material and the need of symmetry and uniformity of the structure for practical reasons. Furthermore, it has to be taken into account that due to manufacturing limitations the design variables are not continuous but discrete, since cross-sections belong to a certain predefined set provided by the manufacturers. Thus the design variables considered are the cross-sectional dimensions of the members of the structure, taken from the circular hollow section (CHS) table. The total number of design variables is eight, since the hangar's elements follow the pattern of the member groups schematically depicted in the truss of Figure 3. The mean value of the modulus of elasticity is 200 GPa and the nominal yield stress is $\sigma_y$=235 MPa (S 235). Equivalent to the vertical uniform load of 0.10 kN, are considered as uniform vertical forces applied at all joints. In addition, a maximum load of 50 kN corresponding to crane is equally distributed at the central sixteen nodes of the roof. The structure is hinged at the boundaries along the transverse direction. Loading conditions are grouped in two categories: (i) dead loads (G) and (ii) live loads (Q). Thus each truss member is checked for actions that correspond to the load combination 1.35G+1.50Q.

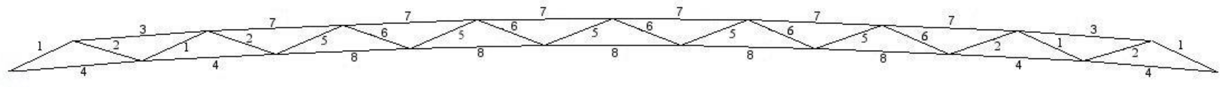


Figure 3: Description of the design variables for the aircraft hangar

In this study three types of behavioural constraints are imposed for the steel type of structures considered: (i) stress, (ii) compression force (for buckling) and (iii) displacement constraints. The stress constraint can be written as follows:

$$\begin{gathered}\sigma_{\max} \leq \sigma_a \\ \sigma_a = \frac{\sigma_y}{1.10}\end{gathered} \tag{10}$$

where $\sigma_{\max}$ is the maximum axial stress of each element group for all load cases, $\sigma_a$ is the allowable axial stress according to Eurocode 3 [11] and $\sigma_y$ is the yield stress. For members under compression an additional constraint is used for buckling:

$$\begin{gathered}\left|P_{c,\max}\right| \leq P_{cc} \\ P_{cc} = \frac{P_e}{1.05} \\ P_e = \frac{\pi^2 EI}{L^2}\end{gathered} \tag{11}$$

where $P_{c,\max}$ is the maximum axial compression force for all load cases, $P_e$ is the critical Euler buckling force taken as the first buckling mode of a pin-connected member, $L$ is the length of each member, while $E$ and $I$ are the material modulus of elasticity and the cross section moment of inertia, respectively. Similarly, the displacement constraints can be written as:

$$|d| \leq d_a \tag{12}$$

where $d$ is the displacement at a certain node or the maximum nodal displacement for all load cases and $d_a$ is the allowable value of the displacement at a certain node or the maximum allowable nodal displacement.

## 4.2 Solution of the multi-objective problem

For the test example considered, two objective functions have been taken into account, the weight of the structure and the standard deviation of the maximum vertical nodal displacement, subject to constraints on stresses, element buckling and displacements imposed by the European design code [11]. A constraint of 500 mm on the maximum deflection is imposed in addition to the stress constraints described in Eqs. (13) and (14). The random variables that have been considered are two per design variable: the external diameter *D* and the thickness *t* of the circular hollow section. Apart from the cross-sectional dimensions of the structural members, the material properties (modulus of elasticity *E* and yield stress $\sigma_y$) and the loads have been considered as random variables, in total 19 random variables are used. The characteristics of the random variables are based on the work by Ellingwood *et al.* [32], where the mean value of the yield stress for the structural steel is 17% higher than its nominal value.

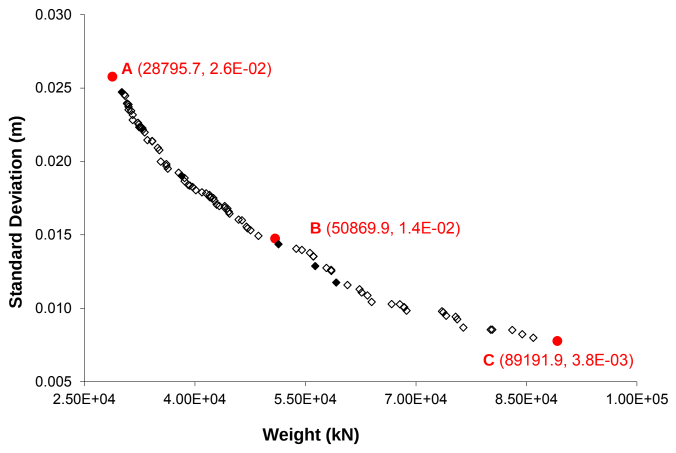


Figure 4: Pareto front curve for the RDO problem

For the solution of the multi-objective optimization problem, with any loss of the generality, the NSES-II(μ+λ) method is employed based on previous experience of the author [33] where $\mu = \lambda = 100$, while a sample size $N_{sim} = 20$ is used. These Monte Carlo simulations are considered adequate when generated with LHS for estimating the mean values and standard deviations in the framework of RDO problems, while the population size is the one employed in the original NSGA-II algorithm [16]. Furthermore, 200 generations are implemented since they were found sufficient to obtain a good quality Pareto front curve in a previous work by the authors [7]. The resultant Pareto front curve for the RDO formulation of Eq. (1) is depicted in Figure 4, with the structural weight on the horizontal axis and the standard deviation of the maximum vertical nodal displacement on the vertical axis. The three designs (A, B and C) are selected from the Pareto front curve for comparative reasons. They represent two extreme designs, corresponding to the two ends of the Pareto front curve, and one compromise design. In particular, design A indicates the Pareto optimum when the weight of the structure is the dominant criterion, design C is the Pareto optimum when the standard deviation of the maximum response is the dominant criterion, while design B indicates a compromise optimum design when both criteria are significant. Comparing design C with the other two designs, it can be seen that design C requires three times more material compared to design A and 40% more compared to design B. On the other hand, the variance of the response for the design C is almost one order of magnitude less compared to that of the other two designs.

## 4.3 Computational efficiency of the DLBA - Death Penalty case

For the purposes of this work a cluster for parallel computing, consisting of 25-nodes is used. Each cluster node consists of the Intel Core 2 Quad Q6600 2.4 GHz with 4 physical cores. For the test example considered both sequential and parallel test runs are carried out. The FETI solution requires partitioning of the FE model in a number of subdomains. For this purpose, a preprocessing step is performed for the domain decomposition of the finite element model upon its creation. Thus, before running the optimization procedure, the geometric configuration of the space truss is decomposed into 2, 5, 10, 25 and 50 subdomains with optimal aspect ratios obtained with TOP/DOMDEC [34]. The number of subdomains depends on the structure, while those selected for the problem at hand were found adequate in a previous work by the author [35]. Upon the creation of each decomposition, every structural element member is assigned to a subdomain on the basis of its distance from the geometrical centre of the subdomain. The minimum distance of each element from the geometric centre of the subdomain designates the subdomain in which the element belongs. The implementation of the proposed dynamic load balancing algorithm requires the computational demands for the FE analysis performed with 1, 2, 5, 10, 25 and 50 subdomains, where each subdomain is allocated to one processor. The computing time required for a single analysis is not affected by any candidate design under consideration, because the truss configuration remains the same during the RDO procedure.

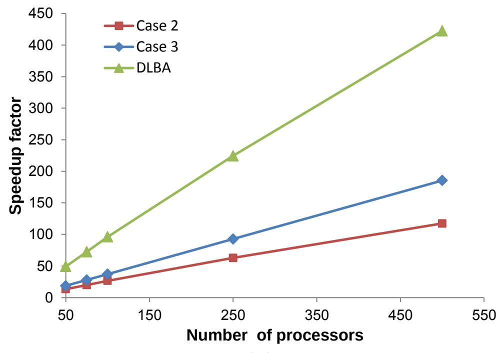
Before assessing the various parallelization schemes implemented in this study, a parametric study is performed with reference to the DE method that is used for the solution of the optimization problem of Eq. (7). The solution of this problem can be performed with any single objective optimization algorithm. The selection of the DE method is based on previous experience of the author [36,37] where it was found very robust. In order to identify the best combination of the parameters for DE algorithm, 30 combinations of the parameters are generated by means of LHS, while for each combination 100 optimization runs are performed in order to calculate the mean and the coefficient of variation with reference to the objective function value. All runs are performed with a cluster of processors comprising $N_{proc}$ = 100 cores. The population size NP of the optimization algorithm is defined in the range of [50, 200], while the probability CR and constant F are defined in the range of [0, 1]. Worth noticing is that for all combinations of the characteristic parameters the best objective function value is finally reached, while COV is below 1%. The characteristic parameters adopted for the implementation of DE method are as follows: population size NP = 195, probability CR = 0.58 and constant F = 0.32.

| **Number of processors** | **Case 1** | **Case 2** | **Case 3** | | **DLBA** | |
|---|---|---|---|---|---|---|
| | | | NSES - $II_{gen}^{syn}$ | NSES - $II_{SS}^{asy}$ | NSES - $II_{gen}^{syn}$ | NSES - $II_{SS}^{asy}$ |
| 50 | $9.69\times10^5$ | $2.74\times10^5$ | $1.97\times10^5$ | $1.97\times10^5$ | $7.42\times10^4$ | $7.35\times10^4$ |
| 75 | $6.55\times10^5$ | $1.85\times10^5$ | $1.31\times10^5$ | $1.31\times10^5$ | $5.04\times10^4$ | $5.00\times10^4$ |
| 100 | $4.88\times10^5$ | $1.38\times10^5$ | $9.85\times10^4$ | $9.85\times10^4$ | $3.79\times10^4$ | $3.76\times10^4$ |
| 250 | $2.07\times10^5$ | $5.81\times10^4$ | $3.94\times10^4$ | $3.94\times10^4$ | $1.63\times10^4$ | $1.59\times10^4$ |
| 500 | $1.11\times10^5$ | $3.11\times10^4$ | $1.97\times10^4$ | $1.97\times10^4$ | $8.65\times10^3$ | $8.56\times10^3$ |

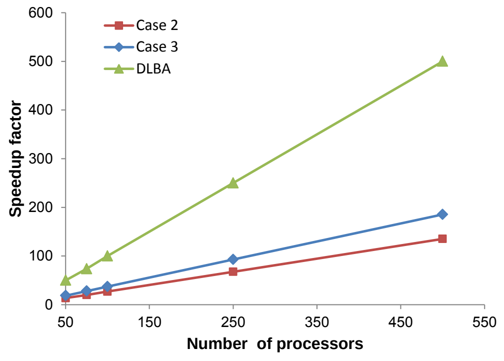
Table 1: Computational performance of NSES-II(μ+λ) for the case of the death penalty constraint handling technique (computing time in secs)

The performance of the NSES-II(100+100) optimization method in serial and parallel computing environments is presented in Table 1. The parallel implementations are performed in five different clusters with 50, 75, 100, 250 and 500 cores. The first three clusters are based on the existing cluster with 100 cores, while the other two are simulated clusters having 250 and 500 cores. In addition to the proposed dynamic load balancing algorithm three other parallelization patterns are examined. *Case 1*: The optimizer is performed in parallel and the solver in serial mode with a skyline solver. The set of the $\lambda_i$ designs that are found to be infeasible define a subgeneration where each design is dealt by one processor. *Case 2*: The optimizer is performed in parallel and the solver is implemented in serial mode with FETI solver, where the infeasible designs are dealt with in a similar fashion as in *Case 1*. *Case 3*: The optimizer and the solver are implemented in parallel mode, where the infeasible designs are dealt independently while the solution part is performed with parallel FETI in more than one processor. In *Case 3* the implementation of FETI in all five different clusters was performed with 50 subdomains (i.e. in 50 processors). The sequential time required for solving the problem at hand is $1.29\times10^7$ seconds with a skyline serial solver and $3.65\times10^6$ seconds with the FETI solver, where almost one order of magnitude reduction of the computing time is achieved with the sequential FETI solver compared to the skyline solver.

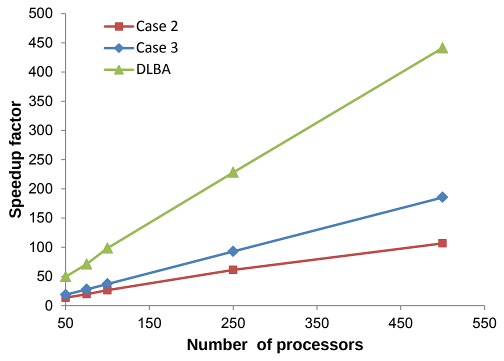
From Table 1 it can be concluded that the proposed procedure managed to reduce the computing time by more than one order of magnitude compared to Case 1, by 4 times compared to Case 2 and by 3 times compared to Case 3. Furthermore, Figure 5 depicts the speedup factors achieved with the implementation of Cases 2 and 3 compared to that of the proposed dynamic load balancing procedure. In this Figure it can be seen that an almost 100% efficiency is achieved with the DLBA, which is much greater compared to that achieved with Cases 2 and 3. Worth mentioning is that the proposed procedure is independent of the population size and the number of Monte Carlo simulations used for the calculation of the statistical properties of the response.



(a)



(b)



(c)

Figure 5: Speedup factors achieved with various parallelization patterns for the $\text{NSES - II}_{gen}^{syn}$ scheme for the case of (a) the death penalty, (b) the superiority of feasible points and (c) the death penalty step size control constraint handling technique

## 5 CONCLUSIONS

- In most cases the design of structures is based on deterministic parameters and is focused on the satisfaction of the deterministically defined design code provisions. A deterministic optimized design is not always a safe design, since such formulations ignore the random properties that affect the structural performance, hence formulations considering uncertainties should be considered. In this work the highly computationally intensive problem of robust structural design optimization is considered, taking into account randomness on both structural capacity and demand. The problem is formulated as a two-objective optimization problem subjected to stress and displacement constraints. A modified nondominated sorting genetic algorithm optimization method is proposed where genetic algorithms are replaced by evolution strategies for the solution of the multi-objective optimization problems. The optimization algorithm is more reliable since it guarantees the feasibility of the Pareto front.

- An important characteristic of the nondominated sorting evolution strategies is that this method works simultaneously with a population of feasible designs only in the search space. This feasibility requirement does not allow for a straightforward implementation of the optimization procedure in parallel computing environments. In this case, a large and variable number of finite element structural analyses can be performed independently and concurrently at each subgeneration step of the optimization procedure. The computational efficiency of the metaheuristic multi-objective optimization algorithms is enhanced with the implementation of parallel domain decomposition algorithms for the solution of the repeated finite element problems and the dynamic exploitation of the available parallel computing resources.
- In this study an efficient dynamic load balancing algorithm is proposed achieving optimum exploitation of the computing resources. The dynamic load balancing problem is dealt with the differential evolution algorithm and the computational efficiency is almost 100%. The reduction of the computational effort achieved is ranging from three times to more than one order of magnitude compared to parallel implementations without the dynamic load balancing. The proposed procedure is independent of the population size and the Monte Carlo simulations used for the calculation of the statistical properties of the response and it is independent on the available computing resources.

## REFERENCES

[1] Tsompanakis, Y., Lagaros, N.D., Papadrakakis (Eds.), M. *Structural Optimization Considering Uncertainties*, Taylor & Francis, 2007.

[2] Park, G.J., Lee, T.H., Lee, K.H., Hwang, K.H. Robust design: An overview, *AIAA Journal 2006*; 44(1): 181-191.

[3] Beyer, H.-G., Sendhoff, B. Robust optimization - A comprehensive survey, *Computer Methods in Applied Mechanics and Engineering 2007*; 196(33-34): 3190-3218.

[4] Lagaros, N.D., Fragiadakis, M. Robust performance based design optimization of steel moment resisting frames, *Journal Earthquake Engineering 2007*; 11(5): 752-772.

[5] Schevenels, M., Lazarov, B.S., Sigmund, O. Robust topology optimization accounting for spatially varying manufacturing errors, *Computer Methods in Applied Mechanics and Engineering 2011*, 200(49-52): 3613-3627.

[6] Youn, B.D., Choi, K.K., Du, L., Gorsich, D. Integration of possibility-based optimization and robust design for epistemic uncertainty, *Journal of Mechanical Design Transactions of the ASME 2007*; 129(8): 876-882.

[7] Lagaros, N.D., Papadrakakis, M. Seismic design of RC structures: a critical assessment in the framework of multi-objective optimization, *Earthquake Engineering and Structural Dynamics 2007*; 36(12): 1623-1639.

[8] Farhat, C., Roux, F.-X. A method of finite element and interconnecting and its parallel solution algorithm, *International Journal for Numerical Methods in Engineering 1991*; 32: 1205-1227.

[9] Farhat, C., Roux, F.-X. Implicit parallel processing in structural mechanics, *Computational Mechanics Advances 1994*; 2: 1-124.

[10] Bitzarakis, S., Papadrakakis, M., Kotsopoulos, A., Parallel solution techniques in computational structural mechanics, *Computer Methods in Applied Mechanics and Engineering 1997*; 148: 75-104.

[11] EC3. *Eurocode 3*: *Design of steel structures*, *Part 1.1: General Rules and Rules for Buildings*. European Committee for Standardisation: Brussels, Belgium, The European Standard EN 1993-1-1: 2005.

[12] Lagaros, N.D., Plevris, V., Papadrakakis, M. Multi-objective design optimization using cascade evolutionary computations, *Computer Methods in Applied Mechanics and Engineering 2005*; 194(30-33): 3496-3515.

[13] Coello Coello, C.A. An updated survey of GA-based multi-objective optimization techniques, *ACM Computing Surveys 2000*; 32(2): 109-143.

[14] Zitzler, E., Deb, K., Thiele, L. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation 2000*; 8(2): 173-195.

[15] Marler, R.T., Arora, J.S. Survey of multi-objective optimization methods for engineering, *Structural and Multidisciplinary Optimization 2004*; 26(6): 369-395.

[16] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 2002*; 6(2): 182-197.

[17] Lagaros, N.D., Fragiadakis, M., Papadrakakis, M. Optimum design of shell structures with stiffening beams, *AIAA Journal 2004*; 42(1): 175-184.

[18] Lee, J., Hajela, P. Parallel genetic algorithm implementation in multidisciplinary rotor blade design, *Journal of Aircraft 1996*; 33(5): 962-969.

[19] Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B. *Evolutionary Algorithms for Solving Multi-objective Problems*, Kluwer Academic Publishers, 2002.

[20] VanVeldhuizen, D.A., Zydallis, J.B., Lamont, G.B., Considerations in engineering parallel multiobjective evolutionary algorithms, *IEEE Transactions on Evolutionary Computation 2003*; 7(2): 144-173.

[21] de Toro Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martín, J.M. PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation, *Parallel Computing 2004*; 30(5-6): 721-739.

[22] Wilson, L.A., Moore, M.D. Cross-pollinating parallel genetic algorithms for multiobjective search and optimization, *International Journal of Foundations of Computer Science 2005*; 16(2): 261-280.

[23] Durillo, J.J., Nebro, A.J., Luna, F., Alba, E. A study of master-slave approaches to parallelize NSGA-II, IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, 2008.

[24] Kipouros, T., Jaeggi, D.M., Dawes, W.N., Parks, G.T., Savill, A.M., Clarkson, P.J., Insight into high-quality aerodynamic design spaces through multi-objective optimization, *Computer Modeling in Engineering and Sciences 2008*; 37(1): 1-44.

[25] Bharti, S., Frecker, M., Lesieutre, G. Optimal morphing-wing design using parallel nondominated sorting genetic algorithm II, *AIAA Journal 2009*; 47(7): 1627-1634.

[26] Fan, S.-K.S., Chang, J.-M. A parallel particle swarm optimization algorithm for multi-objective optimization problems, *Engineering Optimization 2009,* 41(7): 673-697.

[27] Nebro, A.J., Durillo, J.J. A study of the parallelization of the multi-objective metaheuristic MOEA/D. *Lecture Notes in Computer Science 2010*; Volume 6073 LNCS: 303-317

[28] Zhou, Y., Tan, Y. GPU-based parallel multi-objective particle swarm optimization, *International Journal of Artificial Intelligence 2011*; 7(11): 125-141.

[29] Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y.C., Talbi, E.-G., Zomaya, A.Y., Tuyttens, D. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, *Journal of Parallel and Distributed Computing 2011*; 71(11): 1497-1508.

[30] Mezura-Montesa, E., Coello Coello, C.A., Constraint-handling in nature-inspired numerical optimization: Past, present and future, *Swarm and Evolutionary Computation 2011*; 1: 173-194.

[31] Arias-Montano, A., Coello Coello, C.A., Mezura-Montesa, E., Multiobjective evolutionary algorithms in aeronautical and aerospace engineering, *IEEE Transaction on Evolutionary Computation 2012*; 16(50): 662-694.

[32] Ellingwood, B.R., Galambos, T.V., MacGregor, J.G., Cornell, C.A. *Development of a Probability-Based Load Criterion for American National Standard A58*, National Bureau of Standards, Washington, DC, 1980.

[33] Mitropoulou, Ch.Ch., Fourkiotis, Y., Lagaros, N.D., Karlaftis, M.G. Metaheuristics in structural design optimization, in Metaheuristic Applications in Structures and Infrastructures, Amir Hossein Gandomi, Xin-She Yang, Siamak Talatahari and Amir Hossein Alavi (Eds.), Elsevier, 2013.

[34] Sharp, M., Farhat, C. TOPDOMDEC – *A totally object oriented program for visualization, domain decomposition and parallel processing*. User's manual, PGSoft and University of Colorado, Boulder, USA, 1994.

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

[36] Lagaros, N.D., Karlaftis, M.G. A critical assessment of metaheuristics for scheduling emergency infrastructure inspections, *Swarm and Evolutionary Computation 2011*; 1(3): 147-163.

[37] Lagaros, N.D., Papadrakakis, M. Applied soft computing for optimum design of structures, *Structural and Multidisciplinary Optimization 2012*; 45: 787-799.