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)

HIGH PERFORMANCE OPTIMIZATION COMPUTING PLATFORM

Stefanos Sotiropoulos, Georgios Kazakis, 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, Department of Structural Engineering,
School of Civil Engineering, National Technical University of Athens,
9, Heroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece
nlagaros@central.ntua.gr, www.veltion.ntua.gr

Key words: Structural design optimization, final design of real-world structures, shape/topology - parametric optimization, SAP2000, ETABS, SCIA parallel processing.

Abstract: Structural optimization over the past decades matured from an academic theoretical field, to an important tool in the design procedure in various engineering disciplines. Some commercial software applications provide some suites with optimization solutions, but they are focused mostly in the aeronautics, automotive and aerospace industry. High Performance Optimization Computing Platform (HP-OCP) is a software developed by the ISAAR-NTUA and provides a holistic optimization approach for civil engineering structures. More precisely, HP-OCP is a computational suite that has the ability to integrate with several structural analysis and design software and provide optimization solutions. Structural optimization is mainly divided in three groups, sizing (or parametric), shape and topology optimization. All of them are integrated in HP-OCP and the appropriate algorithms are provided in each category. Considering size and shape optimization, the parametric optimization module is developed, in which the design variables of the mathematical formulation can be the dimension of the section properties, the quality of the material, the coordinates of the nodes etc. In this module plenty of derivative-based and derivative-free algorithms are provided like the Projected Quasi-Newton, Constrained Optimization by Linear Approximation, Latin Hypercube Sampling etc. [1]. Considering the topology optimization module [2], the SIMP method is applied and the mathematical algorithms that are implemented are the Optimality Criteria and Method of Moving Asymptotes. HP-OCP was developed in C# programming language, making it a powerful suite that can be integrated with any commercial software that provide Application Programming Interface, batch analysis via XML files or any other type of data exchange format. In the current work the integration of HP-OCP with the SAP2000, ETABS and SCIA Engineering software is presented. Several examples considering parametric and topology optimization problems are examined. Remarkable cost reduction is succeeded in real-world structures, validating in this way the usefulness of HP-OCP not only in the research field but also in applied civil engineering problems.

1. INTRODUCTION

Engineers always strive to design efficient structural systems which must be as economic as possible yet strong enough to withstand the most demanding functional requirements arising during their service life. The traditional trial-and-error design approach is not sufficient to determine economical designs satisfying also the safety criteria. Structural design optimization, on the other hand, provides a numerical procedure that can replace the traditional design approach with an automated one. Automatic numerical optimization algorithms have been developed in the past to meet the demands of structural design optimization. These algorithms can be classified in deterministic and probabilistic approaches. Heuristic and metaheuristic algorithms are nature-inspired or bio-inspired search procedures and belong to the probabilistic class of methods. Modern metaheuristic algorithms are almost guaranteed to an efficient performance for a wide range of optimization problems. Loosely speaking, modern metaheuristic algorithms include genetic algorithms (GA) [3], simulated annealing [4], particle swarm optimization (PSO) [5], ant colony algorithm (ACO) [6], artificial bee colony algorithm [7], harmony search (HS) [8], firefly algorithm [9], pity beetle algorithm (PBA) [10] and many others.

Some software applications in recent years have made these tools accessible to professional engineers, decision-makers and students outside the structural optimization research community. These software applications, mainly focused on aerospace, aeronautical, mechanical and naval structural systems, have incorporated the optimization component primarily as an additional feature of the finite element software package. On the other hand, there is not a holistic optimization approach in terms of final design stage for real-world civil engineering structures such as buildings, bridges or more complex civil engineering structures. The optimized designs, in the case of real-world structures, should rely on accurate modelling and take into account the improvement of the product attributes, such as cost, weight, manufacturability and performance. The implementation of the optimum design formulations to real-world problems requires significant computational effort which can only be addressed with a synergy of cutting edge technologies in the field. Therefore, the tools required for structural optimization basically must serve four purposes (Figure 1): (i) accurate numerical modelling of the structural system; (ii) structural analysis for the calculation of displacements or stresses under various loading conditions, (iii) design procedure for performing the constraints checks imposed by the design codes and/or the design engineers and (iv) search optimization that is the automated procedure for searching for an optimized design that satisfies both the design requirements and economic, manufacturability or performance criteria.

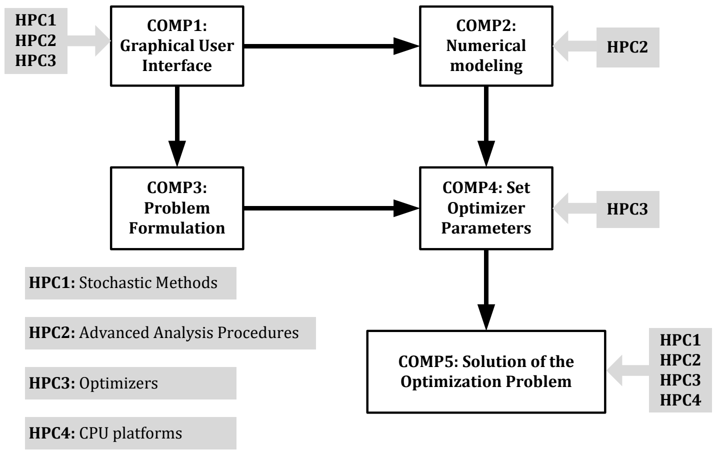
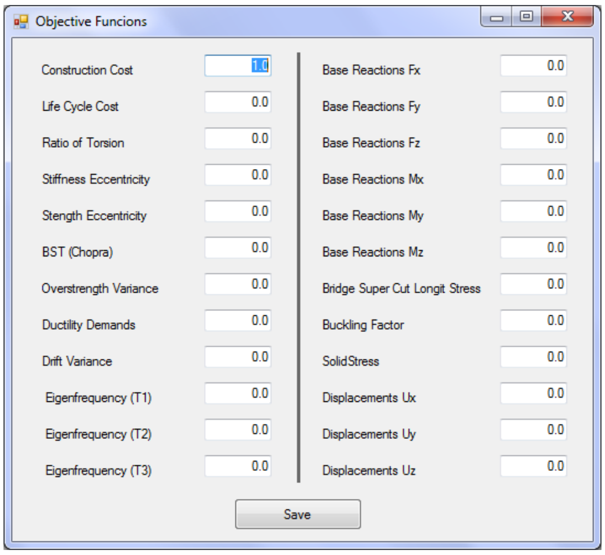

HPC1
HPC2
HPC3
COMP1: Graphical User Interface
COMP2: Numerical modeling
HPC2
COMP3: Problem Formulation
COMP4: Set Optimizer Parameters
HPC3
HPC1: Stochastic Methods
HPC2: Advanced Analysis Procedures
HPC3: Optimizers
HPC4: CPU platforms
COMP5: Solution of the Optimization Problem
HPC1
HPC2
HPC3
HPC4

Figure 1. Flowchart of the HP-OCP computing platform.

The main objective of this work is to present HP-OCP, which is a generic real-world optimum design computing platform for civil structural systems. Utilizing the most developed tools from the above mentioned four categories ensures that the HP-OCP platform can be utilized advantageously. This optimization platform is the result of a 20 year effort to develop a general purpose code for structural analysis and design optimization in the form of a modular standalone code and was developed by OptiStructure [11] and the Institute of Structural Analysis and Seismic Research of the National Technical University of Athens. HP-OCP is based on the next generation computational procedures and it is expected to have a profound impact in optimized design of structural systems, leading to revolutionary changes in civil engineering design practice. HP-OCP computing platform incorporates advanced computational methods that are required for assessing the structural performance within an optimization framework, these methods involve a number of multidisciplinary areas in computational mechanics.

2. PROBLEM DEFINITION

Design optimization refers to the process of generating improved designs in terms of cost, manufacturability or performance. The formulation of the problem has to be defined first that includes the selection of the design criterion (objective function), the design variables and the constraint functions. Structural optimization problems are characterized by various objective and constraint functions that are generally non-linear functions of the design variables. The objective functions supported by the HP-OCP platform are presented in Figure 2. It is allowed to use more than one objective function by introducing a weight coefficient for each one considered. These functions are usually implicit, discontinuous and non-convex.


Objective Funcions

Construction Cost	1.0	Base Reactions Fx	0.0
Life Cycle Cost	0.0	Base Reactions Fy	0.0
Ratio of Torsion	0.0	Base Reactions Fz	0.0
Stiffness Eccentricity	0.0	Base Reactions Mx	0.0
Stength Eccentricity	0.0	Base Reactions My	0.0
BST (Chopra)	0.0	Base Reactions Mz	0.0
Overstrength Variance	0.0	Bridge Super Cut Longit Stress	0.0
Ductility Demands	0.0	Buckling Factor	0.0
Drift Variance	0.0	SolidStress	0.0
Eigenfrequency (T1)	0.0	Displacements Ux	0.0
Eigenfrequency (T2)	0.0	Displacements Uy	0.0
Eigenfrequency (T3)	0.0	Displacements Uz	0.0

Figure 2. Objective functions.

The optimization problems can be expressed in standard mathematical terms as a non-linear programming problem, which in general form can be stated as follows:

$$\begin{aligned} &\text{opt (min/max)} \quad F(\mathbf{s}) \\ &\text{subject to} \quad g_j(\mathbf{s}) \le 0 \quad j=1,\dots,m \\ &\qquad\qquad\quad s_i^L \le s_i \le s_i^U \quad i=1,\dots,n \end{aligned} \tag{1}$$

where **s** is the vector of design variables, F(**s**) is the objective function or the weighted sum of the objective functions to be optimized, $g_j(\mathbf{s})$ are the behavioural constraints imposed by the design codes and the design engineer, s_i^L and s_i^U are the lower and the upper bounds.

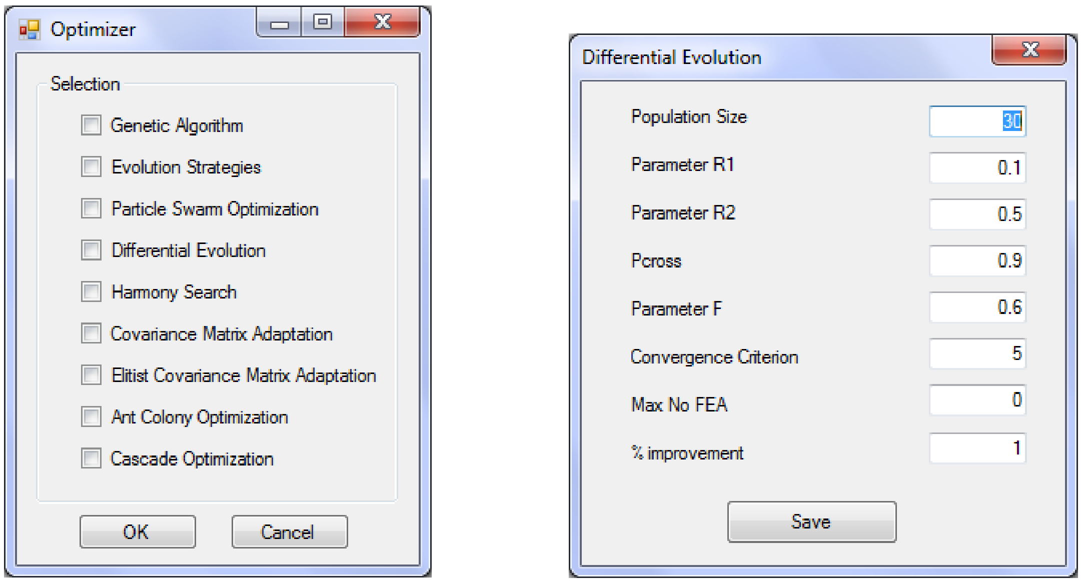
Over the past three decades a remarkable progress was made in the formulation of new structural optimization problems, mainly on a deterministic and, to a lesser extent, on a probabilistic framework as demonstrated in a book by the authors [12]. HP-OCP supports both types of formulations, deterministic and probabilistic ones. In the latter case, in addition to the deterministic requirements, performance criteria are also taken into consideration in connection to various target probabilistic constraints. There are mainly three design formulations that account for the probabilistic system response that are supported by the HP-OCP platform: reliability-based design optimization (RBO), robust design optimization (RDO), and the combined formulation denoted as reliability-based robust design optimization (RRDO). The main goal of RBO formulations is to design for safety with respect to extreme events by determining design points that are located within a range of target probabilities. Thus, in addition to the deterministic constraints, probabilistic constraints are enforced which ensure an acceptable reliability performance. The fundamental principle of RDO is to improve product quality or stabilize performances by minimizing the effects of system's variations without eliminating their causes. In the combined RRDO formulation, additional probabilistic constraints to those of RDO are considered where the maximum probability of violation of behavioural constraints should remain below an allowable probability of violation.

3. THE OPTIMIZATION TOOL

Design optimization is mainly performed by an optimizer, which is an algorithm that is used to search for the "best" design. In HP-OCP various metaheuristic optimization algorithms (MOA) are implemented that appear to be very promising as they have been implemented in various challenging problems with success (Figure 3). Furthermore, the cascade optimization concept was implemented into the computing platform.

3.1 Genetic Algorithms

Genetic algorithms is probably the best-known evolutionary algorithm, receiving substantial attention. The first attempt to use evolutionary algorithms took place in the sixties by a team of biologists [13] and was focused in building a computer program that would simulate the process of evolution in nature. However, the GA model implemented refers to a model introduced and studied by Holland and co-workers [14]. In the basic genetic algorithm, each member of this population will be a binary or a real valued string, which is sometimes referred to as a genotype or, alternatively, as a chromosome. The three main steps of the basic GA: (i) *Initialization*. The first step in the implementation of any genetic algorithm is to generate an initial population. In most cases the initial population is generated randomly. After creating the initial population, each member of the population is evaluated by computing the representative objective and constraint functions and comparing it with the other members of the population. (ii) *Selection*. Selection operator is applied to the current population to create an intermediate one. In the first generation the initial population is considered as the intermediate one, while in the next generations this population is created by the application of the selection operator. (iii) *Generation (Crossover-Mutation)*. In order to create the next generation crossover and mutation operators are applied to the intermediate population to create the next population. Crossover is a reproduction operator, which forms a new chromosome by combining parts of each of the two parental chromosomes. Mutation is a reproduction operator that forms a new chromosome by making (usually small) alterations to the values of genes in a copy of a single parent chromosome. The process of going from the current population to the next population constitutes one generation in the evolution process of a genetic algorithm. If the termination criteria are satisfied the procedure stops otherwise returns to the selection step.



Population Size	30
Parameter R1	0.1
Parameter R2	0.5
Pcross	0.9
Parameter F	0.6
Convergence Criterion	5
Max No FEA	0
% improvement	1

(a) (b)

Figure 3. Optimizer (a) selection of the optimization procedure and (b) the parameters of the selected optimizer

3.2 Evolution Strategies

Evolution strategies (ES) are population based, probabilistic, direct search optimization algorithms gleaned from principles of Darwinian evolution [15]. Starting with an initial population of μ candidate designs, an offspring population of λ designs is created from the parents using variation operators. Depending on the manner in which the variation and selection operators are designed and the spaces in which they act, different classes of ES have been proposed. In the ES algorithm employed in this study, each member of the population is equipped with a set of parameters:

$$\begin{aligned}&\mathbf{a}=[(\mathbf{s}_d,\boldsymbol{\gamma}),(\mathbf{s}_c,\boldsymbol{\sigma},\boldsymbol{\alpha})]\in(I_d,I_c)\\&I_d=D^{n_d}\times R_+^{n_\gamma}\\&I_c=R^{n_c}\times R_+^{n_\sigma}\times[-\pi,\pi]^{n_\alpha}\end{aligned}\tag{2}$$

where $\mathbf{s}_d$ and $\mathbf{s}_c$ are the vectors of discrete and continuous design variables defined in the discrete and continuous design sets D^{n_d} and R^{n_c}, respectively. Vectors $\boldsymbol{\gamma}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\alpha}$ are the distribution parameter vectors taking values in $R_+^{n_\gamma}$, $R_+^{n_\sigma}$ and $[-\pi,\pi]^{n_\alpha}$, respectively. Vector $\boldsymbol{\gamma}$ corresponds to the variances of the Poisson distribution. Vector $\boldsymbol{\sigma}\in R_+^{n_\sigma}$ corresponds to the standard deviations ($1 \le n_\sigma \le n_c$) of the normal distribution. Vector $\boldsymbol{\alpha}\in[-\pi,\pi]^{n_\alpha}$ is related to the inclination angles ($n_\alpha = (n_c-n_\sigma/2)(n_\sigma-1)$) defining linearly correlated mutations of the continuous design variables $\mathbf{s}_c$, where $n = n_d + n_c$ is the total number of design variables.

Let $P^{(t)} = \{\mathbf{a}_1,\ldots,\mathbf{a}_\mu\}$ denotes a population of individuals at the tth generation. The genetic operators used in the ES method are denoted by the following mappings:

$$\begin{aligned} &\mathrm{rec}\ :\ (\mathrm{I_d},\mathrm{I_c})^{\mu} \rightarrow (\mathrm{I_d},\mathrm{I_c})^{\lambda}\ \text{(recombination)} \\ &\mathrm{mut}\ :\ (\mathrm{I_d},\mathrm{I_c})^{\lambda} \rightarrow (\mathrm{I_d},\mathrm{I_c})^{\lambda}\ \text{(mutation)} \\ &\mathrm{sel}_{\mu}^{k}\ :\ (\mathrm{I_d},\mathrm{I_c})^{k} \rightarrow (\mathrm{I_d},\mathrm{I_c})^{\mu}\ (\text{selection, } k \in \{\lambda,\ \mu+\lambda\}) \end{aligned} \tag{3}$$

A single iteration of the ES, which is a step from the population $P_p^{(t)}$ to the next parent population $P_p^{(t+1)}$ is modelled by the mapping

$$\mathrm{opt_{ES}}\ :\ (\mathrm{I_d},\mathrm{I_c})_t^{\mu} \rightarrow (\mathrm{I_d},\mathrm{I_c})_{t+1}^{\mu} \tag{4}$$

At the beginning of the procedure in generation t = 0 the initial parent population $P_p^{(t)}$, composed by μ design vectors, is generated randomly. The next steps correspond to the main part of the ES algorithm, where λ offspring vectors are generated by means of recombination and mutation. D_l is a sub-population with two members selected from the parent population of the current generation $P_p^{(t)}$ which is used by the recombination operator.

3.3 Particle Swarm Optimization

In particle swarm optimization [5], multiple candidate solutions coexist and collaborate simultaneously. Each solution is called “a particle” having a position and a velocity in the multidimensional design space while a population of particles is called a swarm. A particle “flies” in the problem search space looking for the optimal position. As “time” passes through its quest, a particle adjusts its velocity and position according to its own “experience” as well as the experience of other (neighbouring) particles. A particle's experience is built by tracking and memorizing the best position encountered. A PSO system combines local search (through self-experience) with global search (through neighbouring experience), attempting to balance exploration and exploitation. Each particle maintains its two basic characteristics, velocity and position, in the multi-dimensional search space that are updated as follows:

$$\boldsymbol{v}^j(t+1) = w\boldsymbol{v}^j(t) + c_1\boldsymbol{r}_1 \circ \left(\boldsymbol{s}^{\mathrm{Pb},j} - \boldsymbol{s}^j(t)\right) + c_2\boldsymbol{r}_2 \circ \left(\boldsymbol{s}^{\mathrm{Gb}} - \boldsymbol{s}^j(t)\right) \tag{5}$$

$$\boldsymbol{s}^j(t+1) = \boldsymbol{s}^j(t) + \boldsymbol{v}^j(t+1) \tag{6}$$

where $\boldsymbol{v}^j(t)$ denotes the velocity vector of particle j at time t, $\boldsymbol{s}_j(t)$ represents the position vector of particle j at time t, vector $\boldsymbol{s}^{\mathrm{Pb},j}$ is the personal best ever position of the j^{th} particle, and vector $\boldsymbol{s}^{\mathrm{Gb}}$ is the global best location found by the entire swarm. The acceleration coefficients c_1 and c_2 indicate the degree of confidence in the best solution found by the individual particle (c_1 - cognitive parameter) and by the whole swarm (c_2 - social parameter), respectively, while r_1 and r_2 are two random vectors uniformly distributed in the interval [0,1].

3.4 Differential Evolution

In 1995, Storn and Price [16] proposed a new floating point evolutionary algorithm for global optimization and named it differential evolution (DE), by implementing a special kind operator which sought to create new offsprings from parent chromosomes. DE is a relatively novel parallel direct search method which utilizes a population of *NP* parameter vectors $\boldsymbol{s}_{\mathrm{i,g}}$ (i=1,..,*NP*) for each generation *g*, in a recent study [17] a state of the art review on DE is presented. DE generates new vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector corresponds to a better objective function value than a population member, the newly generated vector replaces this member. The comparison is performed between the newly generated vector and all the members of the population excluding the three ones used for its generation. Furthermore, the best parameter vector $\boldsymbol{s}_{\mathrm{best,g}}$ is evaluated in every generation in order to keep track of the progress achieved during the optimization process. According to the variant implemented, a donor vector $\mathbf{v}_{i,g+1}$ is generated first according to:

$$\boldsymbol{v}_{i,g+1} = \boldsymbol{s}_{r_1,g} + F \cdot (\boldsymbol{s}_{r_2,g} - \boldsymbol{s}_{r_3,g}) \tag{7}$$

before the computation of the ith parameter vector $\mathbf{s}_{i,g+1}$. This step is equivalent to the mutation operator step of genetic algorithms or evolution strategies. Integers r_1, r_2 and r_3 are chosen randomly from the interval [1,*NP*] while $i \neq r_1$, r_2 and r_3. *F* is a real constant value, called mutation factor, which controls the amplification of the differential variation ($\boldsymbol{s}_{r_2,g} - \boldsymbol{s}_{r_3,g}$) and is defined in the range [0,2]. In the next step the crossover operator is applied by generating the trial vector $\boldsymbol{u}_{i,g+1} = [u_{1,i,g+1}, u_{2,i,g+1}, \ldots, u_{D,i,g+1}]^T$ which is defined from the elements of the vector $\boldsymbol{s}_{i,g}$ and the elements of the donor vector $\boldsymbol{v}_{i,g+1}$ whose elements enter the trial vector with probability *CR* as follows:

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & \textit{if } \text{rand}_{j,i} \le CR \text{ or } j = I_{rand} \\ s_{j,i,g} & \textit{if } \text{rand}_{j,i} > CR \text{ or } j \ne I_{rand} \end{cases} \tag{8}$$
$$i = 1,2,\ldots,NP \text{ and } j = 1,2,\ldots,n$$

where $rand_{j,i} \sim U[0,1], I_{rand}$ is a random integer from [1,2,...,n] that ensures that $\boldsymbol{v}_{i,g+1} \neq \boldsymbol{s}_{i,g}$. The last step of the generation procedure is the implementation of the selection operator where the vector $\mathbf{s}_{i,g}$, is compared to the trial vector $\boldsymbol{u}_{i,g+1}$:

$$\boldsymbol{s}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g+1} & \text{if } f(\boldsymbol{u}_{i,g+1}) \le f(\boldsymbol{s}_{i,g}) \\ \boldsymbol{s}_{i,g} & \text{otherwise} \end{cases} \tag{9}$$
$$i = 1,2,\ldots,NP$$

3.5 Ant Colony Optimization Algorithm

The ant colony optimization algorithm [6] is a search algorithm inspired by studying the behaviour of ant colonies in their natural environment. In ACO, a colony of artificial ants searches for the best path inside a weighted graph. Consider a population of *m* ants; initially, the ants are randomly positioned on different nodes of the graph. At each step, ant *k* follows a random action choice rule, called random proportional rule, in order to decide which of the nodes will be visited next. While defining the route, ant *k* positioned at node *i*, maintains a memory $\mathcal{M}^k$ containing all nodes previously visited. This memory is used for defining the feasible neighbourhood $\mathcal{N}^k_i$ that contains the nodes that have not been visited by ant *k* yet. The probability with which ant *k,* positioned at node *i*, chooses to move to node *j* is defined as follows:

$$p^k_{i,j} = \frac{(\tau_{i,j})^\alpha \cdot (\eta_{i,j})^\beta}{\sum_{\ell \in \mathcal{N}^k_i} \left((\tau_{i,\ell})^\alpha \cdot (\eta_{i,\ell})^\beta \right)}, \quad \text{if } j \in \mathcal{N}^k_i \tag{10}$$

where $\tau_{i,j}$ is the amount of pheromone between nodes *i* and *j*, α is a parameter controlling the influence of pheromone $\tau_{i,j}$, $\eta_{i,j}$ is a heuristic information that is available a priori, denoting the desirability of the path between nodes *i* and *j* and it given by the following expression:

$$\eta_{i,j} = \frac{1}{d_{i,j}} \tag{11}$$

while in Eq. (11) β is a parameter controlling the influence of the path's desirability $\eta_{i,j}$. According to Eq. (12), the heuristic desirability of moving from node *i* to node *j* is inversely proportional to the distance between nodes *i* and *j*. The probability of choosing a particular connection *i,j* increases with the value of the pheromone trail $\tau_{i,j}$ and the heuristic information value $\eta_{i,j}$. When all ants have completed their routes, the pheromone concentration for each connection between *i* and *j* nodes, is updated for the next iteration *t*+1 as follows:

$$\tau_{i,j}(t+1) = (1-\rho)\cdot\tau_{i,j}(t) + \sum_{k=1}^{m}\Delta\tau_{i,j}^{k}(t), \quad \forall(i,j)\in\mathcal{A} \tag{12}$$

where ρ is the rate of pheromone evaporation, $\mathcal{A}$ is the set of paths (edges or connections) that fully connects the set of nodes and $\Delta\tau^{k}_{i,j}(t)$ is the amount of pheromone ant *k* has deposited on connections it has visited during its tour $\mathcal{T}^k$ and it is given by:

$$\Delta\tau_{i,j}^{k} = \begin{cases} \dfrac{Q}{L(\mathcal{T}^k)} & \text{if connection } (i,j) \text{ belongs to } \mathcal{T}^k \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The coefficient ρ must be less than 1.0 in order to avoid unlimited accumulation of trail [4], while *Q* is constant. In general, connections used by many ants and are parts of short tours, receive more pheromone due to more deposition and less evaporation and are therefore more likely to be chosen by ants in future iterations of the algorithm.

4. GRAPHICAL USER INTERFACE, OPERATING SYSTEM AND DEVELOPMENT

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 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.

4.2 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.

5. APPLICATION TESTS

In order to present all the capabilities of HP-OCP, a wide range of different types of test cases is conducted and the results are illustrated in this section. In particular, two real-world building structures are optimized, the static analysis and design are performed in ETABS v18 and SCIA Engineer software, respectively by means of the *Parametric Design Optimization* Module of HP-OCP. Additionally, two topology optimization problems that are performed with the integration of HP-OCP with SAP2000 are shown, presenting the computational tools of the corresponding *Topology Optimization Module*.

5.1 Parametric Module

In Figure 4(a) a test case of the parametric module that was studied in ETABS.v18 of CSi America, is shown. The structure is a 3-story school concrete building. This building consists of 378 frames, 92 shells, 15 section properties. The plan dimensions are 32.6×8.75 m^2 and the plan area is equal to 275 m^2. The height of the first floor is equal to 3.5 m, while the height of the second and third floor is equal to 3.25m. Three types of concrete material are used: the C20/25, C25/30 and C30/37. The loads applied correspond to the self-weight, dead and live load and earthquake loading. The design combination and the design check are performed according to the Eurocode 2 and 8. A cascade technique is applied where both LHS and PQN algorithms are used and a cost reduction of 14.31% is succeed after 209 iterations.

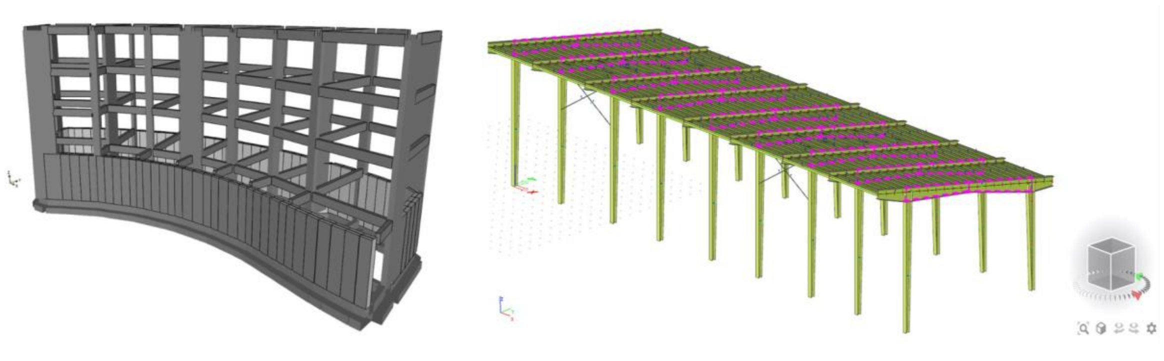

(a) (b)

Figure 4. Parametric optimization by means of HP-OCP integrated with (a) ETABS.v18, and (b) SCIA Engineer v19.

In Figure 4(b) the test case that was studied by means of HP-OCP integrated with SCIA Engineer of the Nemetschek Group, is illustrated. The initial design was provided by Nemetschek and part of the structure to be optimized are the beams on the roof of the structure noted with pink dotted lines. All the beams have the same cross section which is a I cross section with varying height from H_1 (minimum height) in the middle to H_2 (maximum height). The loading combination applied was a combination extracted from the LRFD code containing self-weight, wind and snow loads and the design capacity check is a LRFD verification check as well. The Haunch variable represents the position of the minimum height as a percentage of the beam length. The other design variables represent all the beam cross section dimensions except the web thickness. The algorithm chosen to perform the optimization was the PQN algorithm. Both the objective function as well as the constraint are computed directly by the SCIA Engineer software and the HP-OCP is providing the mathematical algorithm. The final values of the design variables after the optimization procedure can be seen on Table 1. The reduction of the objective function was 6.59% and was found after 94 iterations. The maximum capacity violation of the proposed optimal design is near its limit of 1, more accurate at 0.98.

Table 1. SCIA optimization test case – design variables (dimensions in mm).

Design variables	H_1	H_2	B_1	B_2	t_b	t_s
Initial design	610	915	305	305	16	16
Final design	666	789	263	283	9.6	9.6

5.2 Topology Optimization Module

In the following two topology optimization test cases that are applied with the integration of HP-OCP with SAP2000 are presented. The functionality of the filter scheme, the non-optimized feature and the multiple load case problem is highlighted. More precisely, in Figure 5(a) the bridge problem is illustrated.

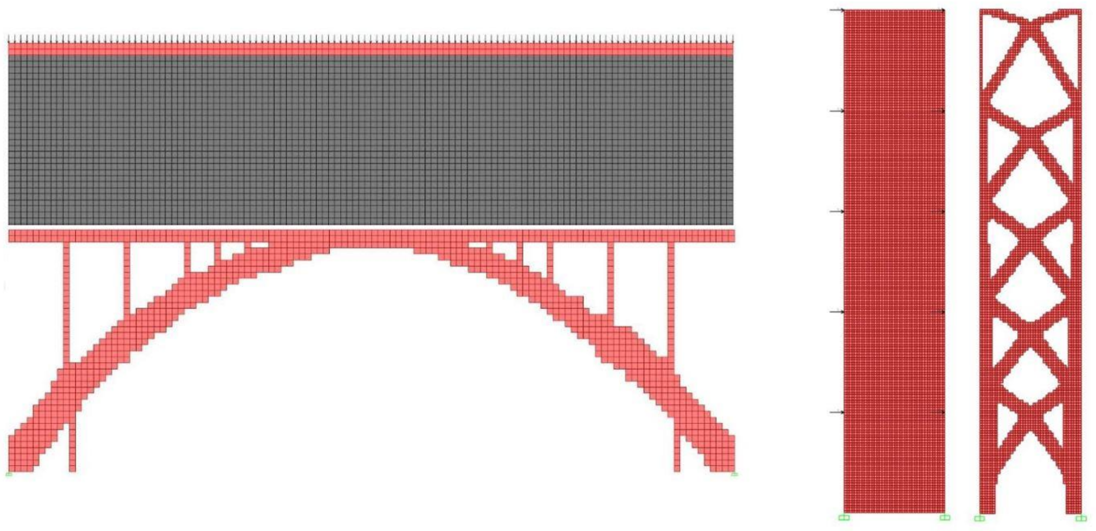

(a) (b)

Figure 5. Topology optimization in SAP2000 for (a) bridge problem and (b) MRF test case

At the bottom left and right the structure is supported and at the top a distributed load is applied. Additionally, at the top of the bridge two rows of elements are predefined in the x direction that it is needed to exist for sure. The structure is discretized with 120 elements in the horizontal direction and 40 in the vertical. 500 different properties are used in order to apply the SIMP method, the volume fraction was set to 0.3 and the radius filter to 3. Finally, the limit of the density that we set for deleting the elements was 0.5. In Figure 5b an MRF test case is presented. The mesh of the structure is 40 elements in the horizontal direction and 200 in the vertical. This is a multiple load case problem as we can see that 5 loads in each side are applied. The rest of the parameters remain the same except, of the limit of the density that we set for deleting the elements, that was set to 0.2.

6. CONCLUSIONS

The features and the capabilities of the structural optimization computing platform developed by OptiStructure and the Institute of Structural Analysis and Seismic Research of the National Technical University of Athens, are reviewed in the current study. The theoretical background of the methods incorporated in the HP-OCP computing platform and their efficiency is given special attention. A special topic in this context is the applicability in real-world civil structural systems; in this direction five real-world application examples are used to illustrate the capabilities of the HP-OCP computing platform.

The optimization platform is the result of a 20 year effort to develop a general purpose code for structural analysis and design optimization in the form of a modular standalone code. HP-OCP is a general-purpose structural optimization code written in C#. Its strength is certainly the broad range of built-in capabilities. It performs structural optimization by a range of eight well-known metaheuristic optimization algorithms while in the case of a probabilistic formulation reliability analysis for structural components and general systems is carried out by the first order reliability method and the Monte Carlo simulation procedure. Therefore, it can be said that HP-OCP offers a bright future in bringing optimization methods into the mainstream of structural engineering practice.

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] N.D. Lagaros, A general purpose real-world structural design optimization computing platform, *Structural and Multidisciplinary Optimization*, 49: 1047–1066, 2014.

[2] S. Sotiropoulos, G. Kazakis, N.D. Lagaros, High performance topology optimization computing platform, *Procedia Manufacturing*, 44: 441-448, 2020.

[3] Goldberg, D.E., *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.

[4] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., Optimization by simulated annealing. *Science*; 220: 671-680, 1983.

[5] Kennedy, J., Eberhart, R., Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*. IV, 1995.

[6] Dorigo, M., Stützle, T., *Ant Colony Optimization*, The MIT Press, 2004.

[7] Karaboga, D., Basturk, B., On the performance of artificial bee colony algorithm, *Applied Soft Computing*; 8(1): 687-697, 2008.

[8] Geem, Z.W., Kim, J.H., Loganathan, G.V., A new heuristic optimization algorithm: harmony search. *Simulation*; 76: 60-68, 2001.

[9] Yang, X.S., *Nature-Inspired Metaheuristic Algorithms*. Frome: Luniver Press, 2008.

[10] N.Ath. Kallioras, N.D. Lagaros, D.N. Avtzis, Pity Beetle Algorithm - A new metaheuristic inspired by the behaviour of bark beetles, *Advances in Engineering Software*, 121: 147-166, 2018.

[11] OptiStructure Ltd. http://optistructure.com/, (January 2021).

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

[13] Barricelli, N.A. Numerical testing of evolution theories, *ACTA Biotheoretica*; 16: 69-126, 1962.

[14] Holland, J., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbour, Michigan, USA, 1975.

[15] Rechenberg, I. *Evolutionsstrategie – Optimierung technischer systeme nach prinzipien der biologischen Evolution*. Fromman-Holzboog, 1973.

[16] Storn, R.M., Price, K.V., Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*; 11: 341-359, 1997.

[17] Ponsich, A., Coello, C.A.C. Differential Evolution performances for the solution of mixed-integer constrained process engineering problems, *Applied Soft Computing*; 11(1): 399-409, 2011.