Table of Contents

List of Figures

List of Tables

Introduction

Since last decades optimization spread everywhere, from engineering design to business planning and from economics to holiday planning.
In almost each of such activities, one tries to achieve one or more objectives or to optimize something as, for instance, time, quality, or profit.
Even in real-world issues, money and time are often limited, one has to find solutions for using these resources in the best way. Certainly, most of these issues have several constraints that make them more diffucult and less immediate.
Thus, mathematical optimization or programming is the study of such plannning and design problem employing some mathematical tools.
Nowadays, for solving most of optimization problems with several efficient search algorithms, computer simulations are essential tool.
In this work, some mathematical tools are described, analyzed and then applied to real-world problems: the metaheuristic algoritms.
The main motivation behind the choice of this branch of optimization tools is that such methods are quite new and reach satisfactory results with a small computational burden. Indeed, such aspect results fundamental because can offer new solutions, and it can lower the economic aspect, that becomes cogent always more.
In this thesis, the attention is focuses on some metaheuristic methods applied on different structures. The achieved results are reported, analyzed and discussed. Starting from a "simple" cantilever beam, these tools are applied until to a large bridge in order to detect and localize the damage.
Then, one of these algorithms has been selected for the reduction of sensors at the bridge deck in a large cable-stayed bridge, maintaing the safety of the structure.
This thesis is organized in five chapters and the topics of the single chapters are:

Chapter 1 – State of the art of the heuristic and metaheuristic algorithms: the main heuristic and metaheuristic algorithms are described.

Chapter 2 – The selected tools developed for structural optimization: some metaheuristic algorithms are chosen and are totally edited in order to frame structural control issues pursued during this research period.
Chapter 3 – Numerical modeling and examples: the numerical examples carried out during the research period are herein presented.
Chapter 4 – Optimization of sensor deployment on the Ting Kau Bridge: the application of a bio-inspired algorithm, in order to find an optimal sensors deployment across a large civil engineering structure for its modal identification is presented.
Conclusions – Some conclusions are drawn.

Chapter 1 State of the art of heuristic and metaheuristic methods for optimization

Most conventional or classic optimization algorithms are deterministic [1]; it means that all the instructions are well-known before its execution. In several deterministic optimization algorithms, the gradient-based algorithms are employed. One of the most notable example of a gradient-based algorithm is the Newton-Raphson algorithm [2]: it utilizes function values and their derivatives and, moreover it fits fine for smooth unimodal problems. Otherwise, whether some discontinuities are present in the objective (fitness) function this algorithm does not work well. Therefore, a gradient-free algorithm is preferred because it does not use any derivative, but only the function values. They are often referred to as stochastic algorithms.

Two main classes of stochastic algorithms can be considered: the heuristic and the metaheuristic. The term *heuristic* means 'to discover by trial and error'. Albeit a notable drawback of the heuristic methods consists in not guaranteeing to achieve the optimal solution, they always reach good sub-optima. Further developments for heuristic algorithms led one to metaheuristic algorithms, wherein the prefix *meta-* means 'higher level' and so they perform better than simple heuristics. The entire class of metaheuristics uses two main features, i.e. randomization and local search.

The first one provides a good approach to move away from local search to the search on the global scale. More in general, any metaheuristic algorithm owns two main factors: intensification and diversification, also named exploitation and exploration. Intensification implies to focus on the search in a local (or specific) region, while diversification implies to generate different solutions. Thus, the global optimality is achievable with the combination of these two major factors. It is worth noticing that metaheuristic algorithms can be classified as population-based and trajectory-based.

1.1 Why the choice of metaheuristic methods?

In optimization, local search methods are algorithms, which first generate an initial candidate. When dealing with the benefits of Genetic Algorithms (GAs), the global search approach will outperform the local search and statements such as using an unlimited population size with unlimited iterations or generations can be handled. A local search algorithm creates a new solution, modifying the old one, in each iteration of its main loop. At this stage, it decides whether to keep the old solution, or to drop a new one and keep going for the next iteration. This idea has a large drawback that it can easily be trapped in a local optimum, hence not the best possible solution. Simulated Annealing and Tabu Search are well-known variants that base their decision on the passed runtime/iteration or search history for mitigating this problem.
Genetic and Evolutionary Algorithms follow another idea, i.e. maintaining a population, a larger set of μ solutions. In each iteration, they create λ new *offspring* solutions from these *parents* and towards these λ, they select the next μ parents. This is an approach, which shields against the premature convergence to local optima, if the population is enough different and allows one to perform a new form of search steps: recombination operators that combine different solutions into new offspring solutions. This approach has been adopted in several global optimization methods such as Estimation of Distribution Algorithms (EDAs) and Ant Colony Optimization (ACO), where a set of solutions is iteratively refined. However, it can be wrong if one uses a huge (unlimited) population size with a large running budget (unlimited iterations or generations) for both global search and local search, and the global search will outperform the local search.
Let first imagine an unlimited population size of a global search method. Of course, the GA will find the global optimum if its population is uniformly randomly initialized. Since the population size of GA is infinite, an infinite time it would be taken to perform the first generation ant it would find the global optimum in the first generation. Hence, at any elapsed finite amount of runtime, it would have only sampled random solutions, i.e. the behavior is indistinguishable from a random sampling process. Good local search methods are likely to outperform random sampling if the meaning of the performance is not reduced to the "runtime until the optimum is discovered" [3]. Certainly, infinite population sizes are an asymptotic and unrealistic assumption. On the other side of the scale, at population size one, the GA becomes a local search itself, so it can hardly outperform local searches. Let now imagine a GA with a finite population, but granted infinite runtime. Assume the search operations employed by the GA are complete and eventually find the possible global optimum solution. Nevertheless, there are some effects that can

stop the efficiency of the GA, for instance the premature convergence. While a metaheuristic method cannot be trapped in a premature convergence because it uses different operations such sums or differences and avoids the derivative.
Of course, premature convergence is a special case, but one cannot rule it out.
Another important issue is the resolution of hard optimization problems within feasible time. GAs are not so fast in term of convergence, while the speed of metaheuristic methods is proved and can be attributed also to the easiness of the calculations [4].
The algorithm is eventually considered good by the idea that the ability of an algorithm to find the global optimum.
Random sampling will find the global optimum eventually, if given enough time. If finding the global optimum was indeed, a criterion for whether an algorithm is good or not, one must show that it does so faster than random sampling. This forces us to abandon assumptions about infinite populations and infinite runtimes.
In conclusion, one could assume that metaheuristics for global optimization necessarily will outperform local optimization algorithms.
In the following of this chapter, some of heuristic and metaheuristic methods are described and the state of the art of such methods is drawn.

1.2 Overview on heuristic methods

In this section, the heuristic algorithms are considered. Such terms derives from the Greek word *heuriskein* (*Εύρίσκω*) which means to find or discover. This word is used in the field of optimization to characterize a specific kind of problem-solving methods. Indeed, there is a wide number and variety of difficult problems, which come up in practice and need to be solved efficiently. Opposing to *exact methods* that ensure to achieve an optimum solution to the problem, heuristic methods only attempt to reach a good, but not necessarily optimum solution. Nevertheless, the time employed by an exact method to find an optimum solution to a complex issue is in a much greater order of magnitude than the heuristic one. Thus, one can conclude that heuristic methods are able to solve real optimization problems [5].

1.2.1 Genetic Algorithms

Genetic Algorithms (GAs) are numerical optimization algorithms inspired by both natural selection and natural genetics [6]. The method is general, and able to be applied to a large range of problems dealt day-by-day. The complex concept that evolution generated the

bio-diversity is a helpful paradigm for solving any complex issue. Thus, the thinking of enlarging the concept of natural selection and natural genetics to other problems is obvious and inspired computer scientist since the early of Sixties. The masterpiece on GAs was written by Holland [7] in 1975 and his motivation was beyond the way of approaching to problem solving. Later in Nineties, de Jong [8] demonstrated that GAs are potentially far more than just a robust method for estimating a series of unknown parameters within a model of a physical system.
A typical genetic algorithm can be summarized in four main steps [6]:

(1) fix a number, or population, of guesses of the solution to the problem,
(2) use a procedure of calculating how good or bad the individual solutions within the population are,
(3) adopt a method for mixing fragments of the better solutions to form new, on average even better solution,
(4) apply a mutation operator to avoid permanent loss of diversity within the solutions.

1.2.1.1 Some applications of Genetic Algorithms

GAs have the capability of solving complex optimization problems, where other methods or tools had difficulty to reach a solution or, worse, they failed. On large-scale problems, some examples within complex spaces riddled with many local optima were dealt with gas pipe layouts among the others. The ability of any genetic tools consists of tackling search spaces with several local optima. Amongst many practical problems and areas to which GAs have been successfully applied, one can mention, image processing, [9], [10], laser technology [11], [12], water networks [13], [14], architectural aspects of building design [15], [16], [17], [18] and structural damage detection [19], [20], [21], [22].

1.2.1.2 Genetic Algorithms operators

Genetic algorithms are initialized with a population of guesses, rather than beginning from a single point within the search space. Such guesses are typically random and spread over the search space. Usually, a genetic algorithm uses three main operators, i.e., *selection*, *crossover* and *mutation* to direct the population towards convergence at the global optimum (either maximum or minimum).

Furthermore, initial guesses are kept as binary encodings or strings of the true variables, although an increasing number of GAs use "real-valued", i.e., base-10, encodings or encodings that have been chosen to replay the natural data structure of the problem. Thus, the initial population is then processed by these three main operators.
First, *selection* attempts to apply pressure upon the population in a similar way of natural selection in biological systems. Thus, poorer performing individuals are discarded and better performing individuals, also called *fitter*, have greater chance of promoting the information they contain within the next generation. *Crossover* allows solutions to swap information in a way that could be compared to that used by a natural organism undergoing sexual reproduction. A method, known as *single point crossover*, is to choose couples of individuals fostered by the selection operator, then randomly choose a single point or locus in the binary strings and finally trade information to the right of the point between the two individuals.
Mutation is used to randomly change (flip) the value of single bits within individual strings and it is used very fairly.
Once selection, crossover and mutation have been applied to the initial population, a new one has formed and the generational counter is increased. Such process of selection, crossover and mutation is carried out until a fixed number of generation have performed or when a convergence criterion has been met.
In the following subsections, these three operators are described in a specific way.

1.2.1.2.1 Mutation

In the natural world, a big amount of processes can create mutation, the simplest being an error during replication. If one thinks about mutation as binary representation, this phenomenon is easy to implement. Indeed, with each new generation the whole population is exchanged, with every bit position in every string visited, and rarely a 1 is turned over a 0 and vice versa. The probability of mutation, P_m, is of the order 0.001, i.e. one bit has the probability to mutate over a thousand. Depending on the problem, the correct setting from P_m changes, however typically is used as $P_m \approx 1/L$ where L is the length of the individual population member:

$$L = \sum_{j=1}^{M} l_j \qquad (1)$$

where l_j denotes the length of a sub-string represented by the M unknowns.

Some authors carried out mutation by visiting each bit position, giving at random 0 or 1 and substituting the current bit with the new chosen value. As there is a 50% probability that the pre-existing bit and the substituted one are identical, mutation occurs only at half the rate suggested by the value of P_m.

1.2.1.2.2 Selection

The basic selection operator is very simple; indeed the best 50% are selected to reproduce, while the remainder has rejected. Such method is practical but it is not the most common. The first reason is that although it permits the best to reproduce, it makes no distinction between "good" or "very good". Furthermore, rather than only permitting poor solutions to go forward to the next generation with a lower probability, it simply deletes them, so reducing the genetic diversity of the population. Amongst selection operators, the more common is the *fitness-proportional* or *roulette wheel*. Adopting this approach the probability of selection increases or decreases in relation to an individual's fitness. The analogy with a roulette wheel occurs because the whole population can be imagined forming a roulette wheel with the size of any individual slot proportional to its fitness. Then the wheel is spun and the "ball" (only figurative) is thrown into. Thus, the probability of the ball coming to rest in any particular slot is proportional to the arc of the slot and so to the fitness of corresponding individual.
The selection mechanism is applied twice in order to select a couple of individuals to undergo (or not) crossover. Such mechanism is carried out until *N*, i.e. the population size, individuals have been selected.
In this context, the type of selection used is denoted by the value of φ, with $\varphi = \dot{r}$ indicating fitness-proportional roulette wheel selection.

1.2.1.2.3 Crossover

The Genetic Algorithm uses single point crossover as the recombination operator. The couples of individuals selected undergo crossover with a probability P_c. A random number R_c is generated in the interval 0-1, and any individual undergoes crossover whether $R_c \le P_c$, otherwise the couples proceed with no crossover. Typically, values of P_c are in the interval from 0.4 and 0.9 and when $P_c = 0.5$ then half the new population is formed by selection and crossover, and half only by selection.

If the crossover is not present, the average fitness of population, f_{ave}, will climb until it equals the fitness of the fittest member, $f_{\max}$. Reached this point, it can only improve via mutation. Moreover, this operator provides a method whereby information for differing solutions can be merged to permit the exploration of new parts of the search space. As described in section 1.2.1.2, single point crossover proceeds by cutting the couple of selected strings at random place and trading the tails to create two child strings. Figure 1.1 clarifies it when the number $R_L = 4$.

Parents	**Children**
1010/0010101	*1010/1111111*
1111/1111111	*1111/0010101*

Figure 1.1 Swapping parents to children (a general representation of the behavior of GAs)

Hence, the new population consists of *N* individuals created by selection and crossover. Therefore, mutation performs on the whole population except the elite number (if the elitism is being applied). When such operation is done, the old population is replaced by the new one the generational counter, *g*, incremented by one.

1.2.2 Ant Colony Optimization & behavior of ants

Ants are social insects that live in organized colonies that can vary from 2 to 25 million. During the operation of foraging, ants communicate or anyway interact in their local environment. In order to communicate each other, ants lay pheromones (or scent chemicals) and thus, each ant can follow the others going through the route beforehand marked with pheromones laid by other ants [23]. When a food source is found, ants "soil" it with pheromones and mark the path to and from the food itself. Initially, the foraging route is simply random, the pheromone concentration varies and then ants follow the one with higher concentration and, exploiting these steps, pheromones grow up because the ants' concentration enhances. The path becomes the favorite when the preferred route is chosen by a wide number of ants. In this way, several favorite routes emerge and so, these ones are more efficient and often the shortest. The interactions among each ant increase the emerging behavior, which exists in an ant colony. The behavior of any individual ant depends to single and local information, such as pheromone concentration, in order to achieve any activity. Albeit no master ant, which oversees the colony and broadcasts in-

structions to other ants is present. Moreover, this behavior is similar to other self-organized phenomena, which occur in several process in nature (i.e., the pattern formation in animals' skins such as tigers or zebras).
Furthermore, the foraging pattern of some ants' species has awesome regularity; in fact, army ants search for food along some regular routes, tilt of about 123°. Thanks to this special behavior, ants choose a different path every day, discovering an area in a couple of weeks [24].

1.2.2.1 Ant Colony Optimization

Ant colony optimization is a bio-inspired algorithm, which inspired scientists and researchers, whose pioneer was Marco Dorigo in 1992 [24]. Since 1992, several variants of this algorithm appeared in the literature.
The basic steps of the ant colony optimization algorithm are shown in Figure 1.2. Two important issues characterize this algorithm: the first concerns the probability of choosing a route, while the second is the evaporation rate of pheromone [25].
For a network routing problem, the probability of ants at a particular node *i* to choose the route from a node *i* to a node j is expressed by the following equation:

$$p_{ij} = \frac{\phi_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n} \phi_{ij}^{\alpha} d_{ij}^{\beta}} \qquad (2)$$

stating either α and β greater than 0 and denoting them as the influence parameters whose typical value is assumed equal to 2 ($\alpha \approx \beta \approx 2$). While ϕ_{ij} indicates the pheromone concentration on the route between i and j, and d_{ij} is the desirability on the same route.
Furthermore, some *a priori* assumptions about the route as the distance s_{ij} is often used and hence $d_{ij} \propto 1/s_{ij}$ that implies a selection of shorter routes due to the shorter travel time and so a higher concentration of pheromone [26].

Ant Colony Optimization

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x}=(x_1,\ldots,x_n)^T$

for $f(\boldsymbol{x}_0)$ *for a routing problem where* $(i,j)\in\{1,2,\ldots,n\}$

Define pheromone evaporation rate γ

while *(criterion)*

  for *loop over all n dimensions (ore nodes)*

  Generate new solutions

  Evaluate new solutions

  Mark better locations/routes with pheromone $\delta\phi_i$

  Update pheromone: $\phi_i \leftarrow (1-\gamma)\phi_i+\delta\phi_i$

  end for

  Find the current best

end while

Output the best results and pheromone distribution

Figure 1.2 Pseudo-code of Ant Colony Optimization Algorithm

Such probability formula denotes that ants usually follow the paths with higher pheromone concentration. Assuming the simplest case, i.e. when $\alpha=\beta=1$, the probability of choosing a path by a group of ants is proportional to the concentration of pheromone on that path. Moreover, the denominator in Equation (2) normalizes the probability, hence is in the range between 0 and 1.

The pheromone concentration can also modify with the time depending on the evaporation of pheromone; this phenomenon is also an advantage because the system avoids the local optima, which could be trapped into. If the evaporation is absent, so the path randomly chosen by the first ant will become the preferred path. For a constant rate γ of pheromone decay or evaporation, one can express the pheromone concentration which usually varies with time exponentially

$$\phi(t)=\phi_0 e^{-\gamma t} \qquad (3)$$

where ϕ_0 is the initial concentration of pheromone, while *t* is the time. If $\gamma t \ll 1$, then $\phi(t)\approx(1-\gamma t)\phi_0$. If one considers the unitary time increment $\Delta t=1$, the evaporation can be approximated by $\phi^{t+1}\leftarrow(1-\gamma)\phi^t$. So, an update and simplified pheromone formula is:

$$\phi_{ij}^{t+1}=(1-\gamma)\phi_{ij}^{t+1}+\delta\phi_{ij}^{t+1} \qquad (4)$$

where $\gamma\in[0,1]$ and it is the rate of pheromone evaporation. The second term in Equation (4) denotes the amount of pheromone deposited at time *t* along a route *i* to j when a

single ant walk through a distance *L*. In most implementations $\delta\phi_{ij}^{t} \propto 1/L$ and if there is any ant on a route the pheromone deposit is zero. Figure 1.3 illustrates a standard problem for ant colony optimization.

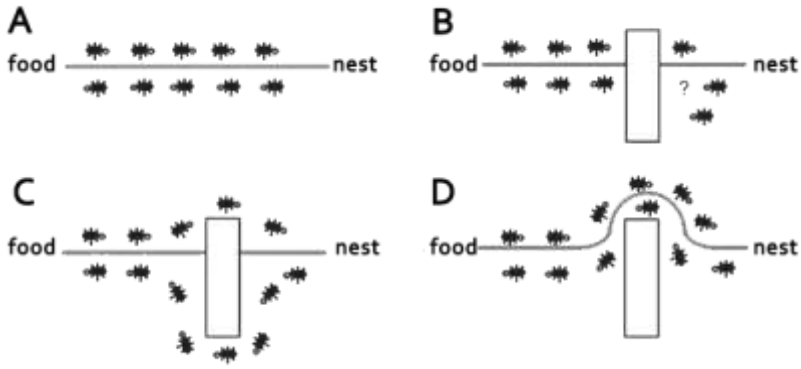


Figure 1.3 Routing performance of ACO

This algorithm has been widely used in different optimization issues which go from the topology optimization [27], to the optimal parameters of tuned mass dampers [28], to photovoltaic systems [29], to structural engineering [30], [31], to design of steel frames [32], and to plane truss optimization [33].

1.2.2.2 Virtual Ant Algorithm

The ant colony optimization has been useful in order to solve strong nonlinear problems as the traveling salesman problem [34], [35], [36], [37], [38], but it can also be expanded to solve generic optimization problem of multimodal functions [39], [40]. The following problem consists in figuring out how ants move on an *n*-dimensional hyper-surface. For instance, the bi-dimensional case can be dealt, and then extended to higher dimensions. Figuring a 2D landscape, any ant has the option to move in any direction or $0° \sim 360°$, but this free movement can produce some troubles. Hence, a solution to update the pheromone at a particular position is to track the history of ant moves and so record the locations, or, alternatively, use a moving neighborhood. Indeed, ants perceive the concentration of the pheromone of their neighborhood at any particular position. In addition, the number of direction where ants can move are limit to 4, i.e., right, left, up and down. This quantized approach facilitates the implementation. Moreover, the objective function is encoded into virtual food, and then ants move directly to the best locations, the ones with

the best food sources. This assumption make the process simpler. Gathering these assumptions, the Virtual Ant Algorithm (VAA) was developed [41], and it has been successfully applied in topological optimization problems in engineering [42], [43].
Finally, it is worthy that ant colony optimization tools are useful for combinatorial and discrete optimization because they capture some features from other stochastic algorithms such the genetics and the simulated annealing.

1.2.3 Particle Swarm Optimization

Kennedy and Eberhart developed particle swarm optimization in 1995 [44], based on the swarm behavior such as birds, fishes and so forth. Since then, PSO has generated great interests and created a new subject called swarm intelligence. PSO has been applied to several optimization issues, computational intelligence, and design/scheduling applications [45], [46], [47], [48], [49]. In literature, more than two dozens of PSO variants exist.
The Adaptive Culture Model hints at what can happen because of the simplest imaginable agents [50]. Indeed, given a large space of possibilities, the population can often manage with multivariate solutions, patterns that solve problems. It is worthy saying that individuals in the culture model are not *trying* to solve issues.
The particle swarm algorithm, here introduced, is viewed in terms of social and cognitive behavior although it is used as a problem-solving method in engineering and computer science. In addition, common versions, which operate in a space of real numbers, are introduced.
The process of cultural adaptation comprises a high-level component, in terms of formation of patterns across individuals, the ability to solve problems, and a low-level component that can be summarized in three main principles [51]: *evaluate*, *compare*, *imitate*. *Evaluate* is the tendency to assess stimuli as positive or negative, attractive or repulsive and so forth; *compare* is the capacity to measure one to each other and imitate only the neighbors who are superior to themselves; finally *imitate* can be found everywhere in the nature, and it is an effective way to learn to do things.
Such three principles of evaluating, comparing, and imitating could be meshed up and help to solve extremely hard problems.

1.2.3.1 Particle Swarm in continuous numbers

The progression of ideas has been from a purely qualitative social optimization algorithm to a model that can been interpreted as qualitative or quantitative. For Kennedy and Eberhart, the particle swarm algorithm searches for optima in the infinitive space of real numbers.

1.2.3.1.1 The Particle Swarm in real-number space

In real-number space, parameters that describe a function can be thought as a point. In engineering applications, system states are points in multidimensional space. Hence, multidimensional space is referred to by various names, depending on the situation. Names include state space, phase space, and hyperspace.
Multiple individuals can be represented within a single set of coordinates, where the measures on a number of individuals will produce a population of points as shown in Figure 1.4.

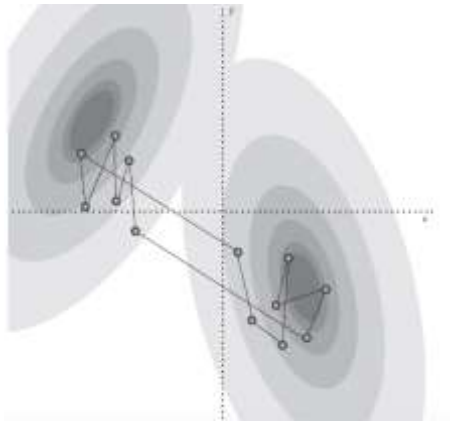


Figure 1.4 Particles in a real-number space

The position of a general particle, *i*, is assigned to an algebraic vector $\mathbf{x}_i$. The number of particles can vary and each vector can assume any dimension. The change of position of a particle is called $\mathbf{v}_i$, velocity. Velocity is a vector which allows to mode the particle from a time step to another:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t) \qquad (5)$$

The particle swarm algorithm samples the search space by modifying the second term in the right hand side of Equation (5).

In the continuous-number particle swarm, a neighborhood is defined for each individual, based on the population array, in turn implemented as a string structure. Thus, the direction of movement is a function of the current position and the velocity, the location of the individual previous best success, and the best position found by any member of the neighborhood:

$$\mathbf{x}_i(t) = f\left(\mathbf{x}_i(t-1), \mathbf{v}_i(t-1), \mathbf{p}_i, \mathbf{p}_g\right) \qquad (6)$$

Figure 1.5 shows a pseudo-code of the PSO algorithm.

Particle Swarm Optimization
Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \ldots, x_p)^T$
Initialize locations $\boldsymbol{x}_i$ *and velocity* $\boldsymbol{v}_i$ *of n particles*
Find $\boldsymbol{g}^*$ *from* $\min\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\}$ *(at t=0)*
while *(criterion)*
  $t = t + 1$ *(pseudo time or iteration counter)*
  for *loop over all n particles and all d dimensions*
  Generate new velocity $\boldsymbol{v}_i^{t+1}$
  Calculate new locations $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}$
  Evaluate objective functions for each particle $\boldsymbol{x}_i^{t+1}$
  Find the current best for each particle $\boldsymbol{x}_i^*$
  end for
  Find the current global best $\boldsymbol{g}^*$
end while
Output the final results $\boldsymbol{x}_i^*$ *and* $\boldsymbol{g}^*$

Figure 1.5 Pseudo-code of Particle Swarm Optimization

1.3 Overview on metaheuristic methods

A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [52], [53]. Metaheuristics are strategies, which convey the search process, and their goal is to efficiently explore the search space in order to find optimal solutions. Furthermore, metaheuristics are approximate and usually non-deterministic. They may incorporate mechanisms to avoid getting trapped in confined areas of the search space. The basic concepts of metaheuristics permit an abstract level description and may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.

1.3.1 Harmony Search

When listening to a beautiful piece of classical music, who has ever wondered if there is any connection between music and finding an optimal solution to a tough design problem such as the water distribution networks or other design problems in engineering? [54]. A group of researchers has found an answer to this question coupling the music to a metaheuristic algorithm. The result is the Harmony Search (HS) algorithm. This algorithm was first developed by Z.W. Geem *et al.* [55] and its advantages and effectiveness have been validated in copious applications. Indeed, after the first appearance, it has been utilized to solve several optimization issues such as function optimization, engineering optimization [56], [57], water distribution networks [58], pipe networks design [59], [60], [61], groundwater modelling [62], reinforced concrete [63], [64] and steel structures [65], [66].
Harmony search is a music-based metaheuristic optimization tool. This algorithm was conceived by the observation that the intent of music is to seek for a perfect state of harmony. One can compare the harmony in the music in finding the optimum in an optimization process. Indeed, the search process during the optimization can be related to a jazz musician improvisation performance. In fact, a musician always pretends to produce a piece of music with perfect harmony. In addition, an optimal solution shall be the best one available to the problem under the given objectives and limited by constraints. Both processes want to produce the best or optimum, respectively. Thus, by learning such similarities between these two processes, one can develop new algorithms. Harmony Search could be seen such as a successful example by transforming the qualitative improvisation process into some quantitative rules by idealization, so turning the beauty and harmony of music into an optimization procedure through search for a perfect harmony, namely, the Harmony Search (HS) or Harmony Search algorithm.

1.3.1.1 Aesthetic quality of music

The pitch (or frequency), the timbre (or quality sound) and the amplitude (or loudness) represent the aesthetic quality of a musical instrument For sake of completeness, the timbre is determined by the harmonic content that is in turn determined by the waveforms or modulations of the sound signal. Even so, harmonics which are generated depend on the pitch range of each particular instrument. Each note has a different frequency and its sound also depends on the temperature and the speed of the air. Thus, adjusting the pitch,

one is trying to modify the frequency. In music theory, pitch p_n in MIDI is usually represented as a numerical scale (a linear pitch space) using the following formula:

$$p = 69 + 12\log_2\left(\frac{f}{440\text{Hz}}\right) \quad (7)$$

or also:

$$f = 440 \times 2^{(p-69)/2} \quad (8)$$

The measurement of harmony is considerably objective as any aesthetic quality considering different pitches that occur at the same time. Anyway, some standard estimation for harmony could be utilized and the frequency ratio, whose innovator was the ancient Greek mathematician Pythagoras, is a good way for such estimations. Figure 1.6 shows that random notes can produce a pleasant harmony.

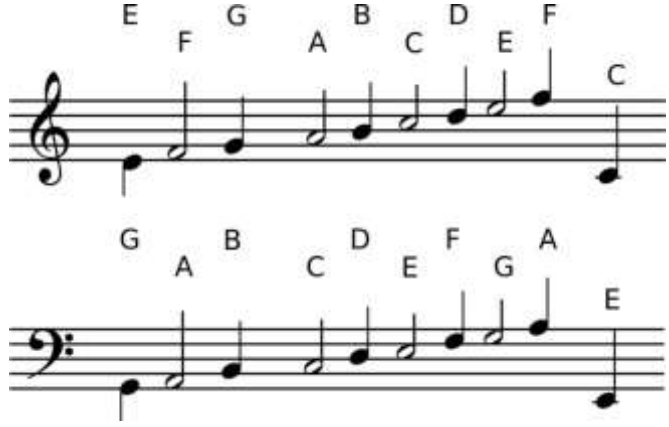


Figure 1.6 Random music notes

1.3.1.2 Harmony Search

The main topic, which can explain in more detail the harmony search is the improvisation process by a musician. Indeed, during the process of improvisation, a musician has three possible choices: (1) playing any famous piece of music (i.e., a series of pitches in harmony) from the memory; (2) playing something similar to a well-known piece (i.e., adjusting the pitch slightly); (3) composing new and random notes. Focusing these three options in the optimization field, one can gather three corresponding features such as usage of harmony memory, pitch adjusting and randomization. The proper usage of the harmony memory becomes fundamental as it is such the choice of the best fit individuals in genetic algorithms. Thus, the best harmonies will be ensured and they carry over to the

new harmony memory. A parameter called harmony memory accepting or considering rate, $r_{accept} \in [0,1]$, can be assigned for use the memory in more effectively. Whether the rate is too low, only a little part of the best harmonies are selected and it may converge slowly. But, whether the rate is extremely high, close to 1, almost all the harmonies are utilized in the harmony memory, then other harmonies are not explored well, leading to potentially wrong solutions. As a consequence, $r_{accept} = 0.7 \sim 0.95$. In order to adjust the pitch in the second component, also the frequency has to be adjusted efficiently. The linear adjustment is utilized, but, theoretically the pitch can be adjusted linearly or non-linearly. Let $\boldsymbol{x}_{old}$ the current solution or pitch, then the new solution (pitch) $\boldsymbol{x}_{new}$ can be written as:

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + b_p \cdot (2 \cdot \text{rand} - 1) \qquad (9)$$

where, rand is a random number drawn from a uniform distribution in the interval $[0,1]$ and b_p is the bandwidth, controls the local range of the pitch adjusting and thus the pitch adjustment in Equation (9) is a random walk.

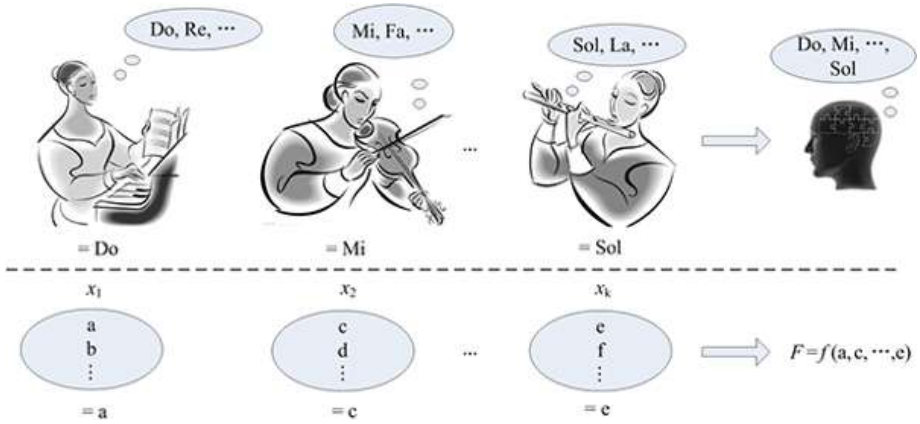


Figure 1.7 Behavior of Harmony Search

Furthermore, the pitch adjustment can be compared to the mutation operator in genetic algorithms. In order to check the degree of the adjustment, a pitch-adjusting rate, i.e. r_{pa} is assigned, but whether r_{pa} is too low, therefore there is a rarely any change. On the contrary, whether the value becomes too high, a non-convergence of the algorithm is not guaranteed. For this reason, in the most of the applications $r_{pa} = 0.1 \sim 0.5$. The third and last component is the randomization, which is useful to increase the diversity of the solutions. Albeit adjusting pitch has a similar duty that is limited only to local pitch adjustment

and thus corresponds to a local search. The usage of this feature can bring the system to explore various regions with high solution diversity in order to find the global optimality. So one writes

$$p_a = p_{lowerlimit} + p_{range} \cdot \text{rand} \quad (10)$$

where $p_{range} = p_{upperlimit} - p_{lowerlimit}$ and rand is a random number in the interval between 0 and 1.

The pseudo-code, shown in Figure 1.8, summarizes the three feature aforementioned. There the probability of the true randomization and the actual probability of pitch adjusting are, respectively

$$\begin{aligned} P_{random} &= 1 - r_{accept} \\ P_{pitch} &= r_{accept} \cdot r_{pa} \end{aligned} \quad (11)$$

Harmony Search Algorithm

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \ldots, x_p)^T$

Generate initial harmonics (real number arrays)

Define pitch adjusting rate (r_{pa}) *and pitch limits*

Define harmony memory accepting rate (r_{accept})

while *(t<Max number of iterations)*

  Generate new harmonics by accepting best harmonics

  Adjust pitch to get new harmonics (solutions)

  if *(rand>* r_{accept}*)*

    Choose an existing harmonic randomly

  else if *(rand>* r_{pa}*),*

    Adjust the pitch randomly within a bandwidth

  else

    Generate new harmonics via randomization

  end if

  Accept the new harmonics (solutions) if better

end while

Find the current best estimates

Figure 1.8 Pseudo-code of Harmony Search

Moreover, harmony search is not a gradient-based search, as genetic algorithms (GA) and particle swarm optimization (PSO) and thus has fewer mathematical requirements and so it can deal with complex objective functions whether linear or nonlinear, stochastic with noise, continuous or discontinuous. The innovation compared to GA is that HS does not use binary encoding and decoding but has multiple solutions vectors, being faster during the process of iterations. In addition, the calibration of the parameters is less sensitive and so one can get quality solutions.

1.3.2 Bee-inspired Algorithms

Improving the search efficiency by using randomness is useful in order to increase the diversity of the solution and thus to avoid being trapped into local optima. The selection of the best individuals is also equivalent to use memory, hence other forms of selection as using chemical messenger, i.e., pheromone, is commonly used by honeybees or also by many other insects [1].
Thus, bee algorithms form a class of metaheuristic algorithms close to the class of the ant colony optimization. Indeed, bee algorithms are inspired by the foraging behavior of the honeybees. In last years, several variants of this algorithm have been developed and formulated. Herein, one can mention the honeybee algorithm (HBA) [67], the virtual bee algorithm (VBA) [68], the artificial bee colony (ABC) [69], [70], [71], the honeybeemating algorithm (HBMA) [72] among the others. These algorithms were used in many different fields such as water resources [73], management of power sources [74], transportation engineering [75], design of truss structures [76] and so forth.

1.3.2.1 Behavior of honeybees

Generally, honeybees live in a colony where they forage and store honey in their constructed colony. The simplest way to communicate for honeybees is by pheromone and "waggle dance". For sake of example, an alarming bee could release a chemical message, the pheromone, in order to simulate attack response in others. Moreover, honeybees are able to communicate the location of the food source once a good one is found and, thus bring some nectar to the hive. In order to communicate their location, they perform the so-called waggle dance as a signal system. Such system changes from a species to another one, but the goal remain unvaried, i.e., recruit more bees by using directional dancing with varying strength. In this way, they can communicate the distance and the direction of the found food resource.

1.3.2.2 Bee algorithms

Since last decades, bee algorithms have emerged as a powerful tool in optimization field, even if it is hard to pinpoint when such algorithms were first formulated. As said in previous section, 2004 and 2005 were two fundamental years: indeed, the honey bee algorithm was firstly formulated by C.A. Tovey with S. Nakrani in order to refine a method to allocate computers among different clients and web-hosting servers [67], while X.S. Yang developed the virtual bee algorithm to solve numerical optimization problems [68].

Furthermore, O.B. Haddad, A. Afshar and M.A. Mariño presented the honeybee-mating optimization algorithm applied to reservoir modelling and clustering. Later in 2006, D. Karaboga and B. Basturk implemented the artificial bee algorithm for numerical function optimization [70].

The main features of bee algorithms are the communication and the location. The first one (also called broadcasting) is the ability of a bee of some neighborhood bees to *know* and follow a bee to the best source, while the second one (or route) is important to complete the optimization issue. The implementation depends on the actual algorithms, and they could differ slightly and change with different variants. Anyway, Figure 1.9 wants to summarize the essence of each bee algorithm.

Bee Algorithms

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x}=(x_1,...,x_n)^T$ *& constraints*
Encode $f(\boldsymbol{x})$ *into virtual nectar levels*
Define dance routine (strength, direction) or protocol
while *(criterion)*
  for *loop over all n dimensions*
    (or nodes for routing and scheduling problems)
  Generate new solutions
  Evaluate the new solutions
  end for
  Communicate and update the optimal solution set
  end if
end while
Decode and output the best results

Figure 1.9 Pseudo-code of Bee Algorithms

1.3.2.2.1 Honeybee Algorithm

In order to maximize the entire nectar intake, forager bees are allocated to different food sources, or flower patches. Thus, such colony has to optimize the overall efficiency of nectar collection, and the allocation of bees depends on several factors, i.e., the proximity to the hive or the nectar richness. This problem is related to the allocation of web-hosting in the internet servers and so inspired researchers to use it.

Consider $w_i(j)$ the strength of the waggle dance of a bee i at time $t=j$, a simple way to find the probability of an observer bee following the dancing bee to foraging is given by:

$$p_i = \frac{w_i^j}{\sum_{i=1}^{n_f} w_j^i} \qquad (12)$$

where n_f states the number of bees in foraging process and *t* is the foraging expedition or the pseudo time. Set *N* the total number of bees, the number of observers is $N - n_f$. Otherwise the exploration probability of a Gaussian type can be defined as $p_e = 1 - p_i = \exp\left[-w_i / 2\sigma\right]$, stating σ the volatility of the colony, which controls the exploration and diversity of the foraging sites. The bees explore randomly whether there is no dancing, i.e. no food found, then $w_i \to 0$ and $p_e = 1$.

In other versions of the algorithm, for example dealing a discrete problem such the job scheduling, a forager bee perform waggle dance with a duration $\tau = \gamma f_p$ where f_p denotes the profitability (related to the objective function) of the food site, while γ is a scaling factor. Along this, the rating of each route is ranked dynamically and the preferred path is the one with the highest number of bees. Dealing a routing problem, Equation (13) describes the probability of choosing a route between any two nodes

$$p_{ij} = \frac{w_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n} w_{ij}^{\alpha} d_{ij}^{\beta}} \qquad (13)$$

where α and β, both greater than 0, are the influence parameters, w_{ij} is the dance strength along route *i* to *j*, and d_{ij} is the desirability of the same route.

1.3.2.2.2 Virtual Bee Algorithm

For solving both continuous and discrete problems, the virtual bee algorithm can be utilized. This algorithm has some similarity to the particle swarm optimization, indeed in VBA the continuous fitness (objective) function is encoded as virtual nectar, while the solutions, i.e. the decision variables, are the locations of such nectar. The different activities, as the waggle dance, are combined with the nectar concentration as the *fitness* of the solutions. So one can recognize the difference for a maximization problem and for a minimization problem. In the first case, the objective function can be considered as virtual nectar, while in the second one, the nectar can be thought in such way that the minimum value of the objective function coincides to the highest nectar concentration.

Whereas, dealing discrete problems, the objective function is strictly linked with the profitability of the nectar explorations that is in turn linked to the waggle dance of the forager

bees. Here, the virtual bee algorithm has a similarity to the honeybee algorithm. However, the VBA has a broadcasting ability of the current best, which becomes the fundamental difference from other bee algorithms. Thus, the current best location appears *known* to every bee so that this algorithm is more efficient. In this way, time is saved because forager bees do not come back to the hive for telling other onlooker bees via *waggle dance*. A similar feature is also used in particle swarm optimization algorithms, especially in accelerated-PSO algorithms.

1.3.2.2.3 Artificial Bee Colony Optimization

The artificial bee colony optimization algorithm was widely studied concerning unstrained optimization problems and its extension.
In the artificial bee colony algorithm, the bees are gathered in a colony and they are divided into three different groups: employed or forager bees, onlooker or observer bees and scouts. Furthermore, for each food source, only one bee is employed. It means that the number of employed bees and food sources is the same. Indeed, the employed bee of a discarded food site automatically becomes a scout for searching new food sources randomly. During the search, employed bees share information with onlookers in a hive, thus onlookers can choose a food source to forage. Differently from the honeybee algorithm, bees are more specialized in ABC.
Let $f(\boldsymbol{x})$ an objective function, it can be encoded as $F(\boldsymbol{x})$ representing the amount of nectar at location $\boldsymbol{x}$, so the probability P_i of an onlooker bee choose to go to a preferred food source located in $\boldsymbol{x}_i$ can be expressed as

$$P_i = \frac{F(\boldsymbol{x}_i)}{\sum_{j=1}^{S} F(\boldsymbol{x}_j)} \qquad (14)$$

where *S* denotes the number of food sources. In fact, at a particular food source, the intake efficiency is led by F/T where *F* is the amount of nectar and *T* is the time spent at the food source. The bees at this location will move on randomly to explore new locations whether a food source is foraged at a given number of explorations without improvement. Several applications have been carried out recently in the last few years in different fields as the combinatorial optimization, job-scheduling, web-hosting allocation and engineering design optimization. In civil engineering, different applications were studied by ABC algorithm, such as in [77], [78], [79] and [80].

1.3.3 Firefly Algorithm

The firefly algorithm (FA) is based on the behavior and patterns of fireflies [1]. This algorithm was proposed for continuous optimization and it was applied in several fields such as structural optimization or image processing. Essentially, FA is based on the three rules: the fireflies are assumed to be unisex so that one among them will be attracted to other fireflies regardless the sex, the attractiveness of a firefly is related to its brightness and decreases with distance and the brightness of a firefly is determined by the landscape of the objective function [81].

The fireflies' movement is determined by their brightness. Furthermore, whether the maximum is achieved and there are two fireflies, then the less bright moves toward the brighter one. While, whether there is no brightness they move randomly. The search of the optimal point, i.e., the global minimum or maximum of a known objective function, is carried out by the objective function itself, not depending from the calculation of its gradient. Unlike classical gradient methods, the firefly algorithm does not introduce errors when either the variables are discrete or the function is not differentiable. Particularly, the tool depends only on few parameters and needs a very moderate computational burden relatively to other metaheuristic algorithms, due to the simple operations to solve, such as sums and differences.

The procedure starts from a defined population (*P*) of fireflies of size *NP*, which belongs to a well-defined existence domain, depending on the dimension d of the problem. The initial locations of these fireflies $\mathbf{x}_i^{(P)}$ ($i = 1, 2, ..., NP$) are uniformly distributed in the entire search space. Each individual of the population is taken as a possible candidate to form the next generation. The distance between any two fireflies $\mathbf{x}_i^{(P)}$ and $\mathbf{x}_j^{(P)}$ is calculated as their Cartesian distance

$$r_{ij}^{(P)} = \| \mathbf{x}_i^{(P)} - \mathbf{x}_j^{(P)} \| = \sqrt{\sum_{k=1}^{d} (x_{i,k}^{(P)} - x_{j,k}^{(P)})^2} \qquad (15)$$

where $x_{i,k}^{(P)}$ is the *k*-th component of the vector $\mathbf{x}_i^{(P)}$ of the *i*-th firefly.

Two important issues are the variation of the light intensity and the formulation of the attractiveness. The light intensity is represented by the value of the cost function evaluated at the current point of the solution space. If the objective is to minimize the fitness function, then the current firefly should move toward a less bright one. The attractiveness is defined as

$$\beta_{ij}^{(P)}(r_{ij}^{(P)}) = \left(1-\beta_{\min}\right)e^{-\gamma r_{ij}^{(P)2}} + \beta_{\min} \qquad (16)$$

where γ is the light absorption coefficient, $\beta_{\min}$ the minimum attractiveness, and $\left(1-\beta_{\min}\right)$ the attractiveness at $r_{ij}^{(P)} = 0$. Hence, the movement of a firefly $\mathbf{x}_i^{(P)}$ that is attracted to another firefly $\mathbf{x}_j^{(P)}$ which is less bright, thus more attractive, is given by

$$\mathbf{x}_i^{(P+1)} = \mathbf{x}_i^{(P)} + \beta_{ij}^{(P)}\left(\mathbf{x}_j^{(P)} - \mathbf{x}_i^{(P)}\right) + \zeta\boldsymbol{\kappa}_i^{(P)} \qquad (17)$$

where the second term in the right hand side is due to the attraction and the third one is the randomization term, with ζ the randomization parameter and $\boldsymbol{\kappa}_i^{(P)}$ the vector of random numbers drawn from a Gaussian distribution.

From any large number of fireflies, the convergence of the algorithm is achieved. Indeed, as the iterations of the algorithm proceed, the fireflies converge to the local optima. Comparing the best solution among the optima, the global optima are eventually obtained.

The described algorithm is summarized in the pseudo-code in Figure 1.10.

Firefly Algorithm
Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \ldots, x_d)^{\mathrm{T}}$
Generate initial population of fireflies $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, n)$
Light intensity I_i *at* $\boldsymbol{x}_i$ *is determined by* $f(\boldsymbol{x}_i)$
Define light absorption coefficient γ
while *(t<MaxGeneration)*
for $i = 1$ *:n all n fireflies*
  for $j = 1$ *:n all n fireflies (inner loop)*
  if *(*$I_i < I_j$*), Move fireflies i towards j;* **end if**
    Vary attractiveness with distance r via $\exp[-\gamma r]$
    Evaluate new solutions and update the light intensity
  end for *j*
end for *i*
Rank the fireflies and find the current global best $\boldsymbol{g}_*$
end while
Postprocess results and visualization

Figure 1.10 Pseudo-code of Firefly Algorithm

Certainly, the performance and the accuracy of the method is based on two crucial steps: (1) the selection of the optimization variables and (2) the formulation of objective function.

This algorithm has been largely used for solving different optimization issues civil engineering field as in damage localization [82], [83], in footbridges design [84], in sensors location[85], [86], [87], in design of tower structures [88].

1.3.4 Echolocation of bats & Bat Algorithm

Bat algorithm [89], which was firstly introduced by Yang [90], has been promising efficiency for global optimization.

1.3.4.1 Behavior of microbats

Bats are the only mammals with wings, and they have advanced capability of echolocation. More than 1000 species of bats exist in nature and they have a size range from 2g (the tiny bumblebee bat) to 1kg (the giant bat with a wingspan of about 2m). Most of bats uses the echolocation to a certain degree. Microbats use echolocation extensively, in contrast to megabats, which do not. Furthermore, most of microbats are insectivores and use a type of sonar, i.e. the echolocation, which helps them to detect prey, avoid obstacles and also locating their roosting crevices in the dark. This specie of bats can emit very loud sound pulses and thus listen to the echo that bounces off from the surrounding objects. Species by species, pulses of bats change in properties and a correlation with the hunting strategy is found. Most part of bats use short, frequency-modulated signals while others more often use constant-frequency signals for echolocation. Each species has a specific bandwidth and often increases by using more harmonics. Furthermore, researchers show that microbats employ the time delay from emission and detection of the echo, the time difference between their two ears, and also the loudness variations of the echoes to build up three dimensional scenario of the surrounding [1]. Indeed, all the studies carried out, shown that bats are able to discriminate targets by the variation of the Doppler Effect, induced by the wing-flutter rates of the target insects.

1.3.4.2 Acoustic of echolocation

Most bat species fall their range of frequencies between 25kHz and 100kHz, albeit among them, any can emit higher frequencies up to 150kHz. Each ultrasonic burst typically last about 5÷20ms, and microbats emit 10÷20 sound bursts in a second. Such pulse emissions hasten to 200 pulses per second during the hunt, especially whether the prey is close. This particular feature shows the awesome ability in signal processing power by bats.
As the speed of sound in air is usually $v = 340\,\mathrm{m/s}$ at room temperature, the wavelength λ of the ultrasonic sound bursts with a constant frequency (*f*) is expressed by the following equation

$$\lambda = \frac{v}{f} \qquad (18)$$

that for the typical frequency range from 25kHz to 150kHz, is in the range of 2mm to 14mm.

Furthermore, the emitted pulse may be louder than 110dB, and so, fortunately, they are in the ultrasonic region. Hence, the loudness varies towards the prey.

Such echolocation behavior of microbats can be formulated as associated with the objective function to be optimized, and it permits to formulate algorithms.

1.3.4.3 Bat Algorithm

Idealizing the echolocation features of microbats above described, one can develop various bat-inspired algorithms. First of all, three approximate rules can be assumed: first, each bat uses echolocation to sense distance, and has the ability to recognize the difference between food/prey and background barriers; second, bats fly randomly with velocity $\boldsymbol{v}_i$ at position $\boldsymbol{x}_i$ with a fixed frequency $f_{\min}$ (or wavelength λ), varying wavelength (or frequency *f*) and loudness A_0 to search for prey. They can automatically adjust the wavelength (or the frequency) of their emitted pulses and also adjust the rate of pulse emission $r \in [0,1]$, which depends on the proximity of their target; third, although the loudness can vary in many ways, the loudness varies from a large positive A_0 to a minimum value $A_{\min}$ [92].

Then, another simplification is assumed, that is no ray tracing in estimating the time delay and three-dimensional topography. This feature reveals interesting for the application in computational geometry. For sake of simplicity, the following approximations are also used. Thus, the frequency *f* in an interval $[f_{\min}, f_{\max}]$ is equal to an interval of wavelength $[\lambda_{\min}, \lambda_{\max}]$.

Set a problem, any wavelength for the easiness of implementation can be employed. Thus, in the current implementation, the interval (or range) can be adjusted by adjusting the frequencies (or wavelengths). The detectable interval, or the largest wavelength, ought to be chosen such that it could be comparable to the size of the domain of interest, and then toning down to smaller ranges. Moreover, not all wavelengths can be necessarily used, but also vary the frequency while fixing the wavelength λ . This is due because λ and *f* are related, since λf is constant.

Another important assumption is that $f \in [0, f_{\max}]$; indeed it is well-known that higher frequencies have short wavelengths and travel a shorter distance. For bats, the typical ranges are few meters and the rate of pulse can be in the interval $[0,1]$ where 0 means no pulse, while 1 means the maximum rate of pulse emission. Figure 1.11 summarizes a pseudo-code of the Bat Algorithm (BA).

Bat Algorithm
Initialize the bat population x_i $(i = 1,2,\ldots,n)$ *and* v_i
Initialize frequencies f_i, *pulse rates* r_i *and the loudness* A_i
while *(t<Max number of iterations)*
  Generate new solutions by adjusting frequency,
  and updating velocities and locations/solutions
  if *(rand* $> r_i$*)*
    Select a solution among the best solutions
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if *(rand* $< A_i$ & $f(x_i) < f(x_*)$
  Accept new solutions
  Increase r_i *and reduce* A_i
  end if
  Rank the bats and find the current best x_*
end while

Figure 1.11 Pseudo-code of Bat Algorithm

This algorithm has been employed in different fields, which go from economics to sport training sessions. Some interesting applications in civil engineering were carried out in last few years [94], [95].

1.3.4.3.1 Movement of virtual bats

Since in simulations virtual bats are employed, the rules that define and update their position $\boldsymbol{x}_i$ and velocities $\boldsymbol{v}_i$ in a *d*-dimensional search space. Following equations describe the new solutions $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ at time step *t*

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right)\beta \qquad (19)$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + \left(\boldsymbol{x}_i^t - \boldsymbol{x}_*\right) f_i \qquad (20)$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t \qquad (21)$$

where $\beta \in [0,1]$ denotes a random vector from a uniform distribution. Herein, $\boldsymbol{x}_*$ is the current global best location (solution), located by comparing each solution among the *n* bats. Denoting the product $\lambda_i f_i$ the velocity increment, one can use f_i (or λ_i) to

adjust the velocity change, whereas fixing the other factor λ_i (or f_i), concerning on the type of problem of interest. In most of implementation, $f_{\min} = 0$ and $f = O(1)$ which depends on the domain size of the problem. First of all, each bat has a random frequency assigned which falls into $\left[f_{\min}, f_{\max} \right]$.

For the local search part, once a solution is chosen among the current best ones, a new solution is then generated for each bat, using random walk

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + \iota A^t \qquad (22)$$

where $\iota \in [-1,1]$ and it is a random number, while $A^t = < A_i^t >$ is the average loudness of each bat at this time step.

The update of the velocities and the positions of bats has some similarity to the procedure in the standard PSO. Indeed, BA can be considered as a balanced combination of the PSO and the intensive local search controlled by the loudness and pulse rate [93].

1.3.4.3.2 Loudness and pulse emission

Moreover, accordingly as the iterations proceed, the loudness A_i and the rate r_i have to be updated. Generally, when the bat has found its prey, the loudness decreases while the rate of pulse emission increases, so the loudness can be chosen as any desired value. For sake of simplicity, one can assumes $A_0 = 1$ and $A_{\min} = 0$; when $A_{\min} = 0$ that means that a bat has just found the prey, thus is no emitting any sound temporarily. So one can write

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0 \left[1 - \exp(-\gamma t) \right] \qquad (23)$$

assuming both α and γ as constants. Furthermore, for any $0 < \alpha < 1$ and $\gamma > 0$, one has

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as } t \to \infty \qquad (24)$$

In the simplest case, the parameters (i.e., α and γ) are assumed to be equal, but the choice of parameters require some experiments.

1.3.5 Cuckoo Search & cuckoo breeding behavior

Cuckoos are fascinating birds, first because of the beautiful sounds, which they emit and for their aggressive reproduction strategy [96]. Some cuckoos species as the *Ani* and the

Guira lay their own eggs in communal nests, albeit they may remove other cuckoos eggs in order to enlarge the hatching probability of their own eggs. Quite a number of species hires the obligate brood parasitism, by laying their eggs in the nests of other host birds, which usually are other species. Moreover, brood parasitism can be subdivided into three basic types, i.e. intraspecific brood parasitism, cooperative breeding and nest takeover [97]. Some host birds are in conflict with the introducing cuckoos and whether a host bird recognize that the eggs are not its own, thus it will get rid of them or will abandon the nest and will build a new one in another place. Anyway, some cuckoos species, as the *Tapera*, have evolved and their female exemplars are often specialized in the imitation in color and pattern of the eggs of a chosen host species. Therefore, the probability of their eggs being abandoned reduces and, consequently the reproductively grows up.
Besides, also the timing of egg-laying becomes interesting. Indeed, parasitic cuckoos often choose nests where the host bird has laid its own eggs and generally, the cuckoo eggs hatch slightly earlier than their host eggs. Once the cuckoo chick is hatched, the first action by the host bird is to evict other eggs out of the nest and that increases the chance of provided food by its host bird. In addition, cuckoo chicks are able to mimic the call of host chicks to gather more feeding opportunity [96].

1.3.5.1 Cuckoo Search

Three idealizing rules can describe the Cuckoo Search very well [98]: first, each cuckoo lays eggs one-by-one and dumps its egg in a random nest previously chosen; second, the best nests, i.e. ones with high-quality eggs, will carried over to the following generations; third, the number of available host nests is fixed and the egg laid by a cuckoo has a probability to be discovered by the host that falls in the interval between 0 and 1 $\left(p_a \in [0,1]\right)$

The third assumption could be approximated by a fraction p_a of the *n* host nests are replaced by new ones.
Dealing a maximization problem, the fitness of a solution is proportional to the value assumed by the objective function. Assuming an implementation point of view, one can imagine that each egg in a nest represents a solution, while each cuckoo can lay only one egg so only one solution, the goal is to employ the new and potentially better solutions to replace the *not-so-good* ones in the nests.

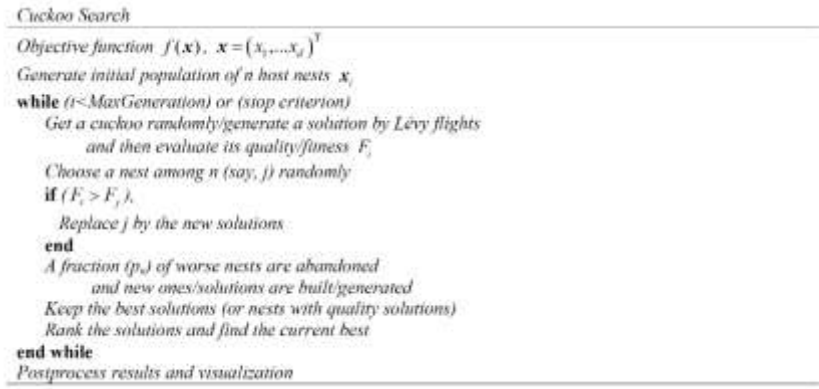


Figure 1.12 Pseudo-code of Cuckoo Search

Thus, based on these three rules, Figure 1.12 summarizes the basic steps of Cuckoo Search (CS).

Let $\boldsymbol{x}^{(t+1)}$ a new solution for a cuckoo *i*, a Lévy flight is performed

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda) \qquad (25)$$

where $\alpha > 0$ states the step size, related to the scales of the problem of interests. In most of implementations, $\alpha = O(L/10)$ where *L* denotes the characteristic scale of the problem of interest. Anyway, Equation (26) represents the stochastic equation for a random walk. Generally, a random walk is a Markov chain whose next location depends on the current location and the transitions probability. Moreover, the product $\oplus$ indicates entrywise multiplications.

A Lévy distribution fits well the Lévy flight, which provides a random walk

$$\text{Lévy} \sim u = t^{-\lambda}, \quad (1 < \lambda \le 3) \qquad (26)$$

which has an infinitive variance and an infinitive mean [99]. Thus, each step forms a random walk process with a power-law step-length distribution with a heavy tail [100]. Some of the new solutions should be generated by Lévy walk about the best solution obtained.

There are some significant differences between CS and hill-climbing optimization technique [101], because at the first sight they seem similar. Nevertheless, first CS is a population-based algorithm and uses some sort of elitism, second, the randomization in CS is more efficient and third, the number of parameters in CS to be chosen is fewer than PSO or GA. Therefore, it is potentially more generic to adapt to a wider class of optimization

problems, such as in the structural optimization [102], [103], [104], in seismic engineering [105], in problems that deals sensor networks [106], [107].

1.3.6 Monkey Algorithm

The Monkey Algorithm (MA), originally developed by Zhao and Tang [108], is a population-based algorithm in the class of metaheuristics. This method derives from the simulation of mountain-climbing processes of monkeys. First, let assume that there are many mountains in a given field (i.e., in the feasible space of the optimization problem) for finding the mountaintop (i.e., in optimization the maximum/minimum value of the objective function), monkeys are climbing up from their current positions (such process is called the climb process). When it gets to the top of the mountain, for each monkey becomes natural to have a look around and to find out if there is the presence of other mountains around it higher than its present whereabouts. If the answer is positive, any monkey jumps somewhere of the mountain observed by it from the actual position (such process is called the watch-jump process), and then it repeats the climb process until it achieves the top of the *new* mountain. After repetitions of the climb and the watch-jump processes, each monkey finds a locally maximal mountaintop around its starting point. Hence, for finding a much higher mountaintop, it is natural for each monkey to somersault to a new search domain (such process is called somersault process). Figure 1.13 illustrates the processes above described.

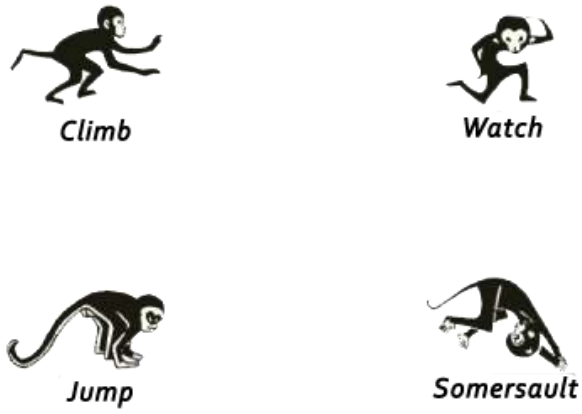


Figure 1.13 Processes of Monkey Algorithm

After a number of repetitions of the climb process, the watch-jump process, and the somersault process, the highest mountaintop found the population of monkeys is reported as the optimal value.

In the MA, the purpose of the somersault process is to make monkeys find new search domains without being trapped into local optima. In addition, the time consumed by the MA mainly lies using the climb process to search local optimal solutions. The essential feature of such process is the calculation of a pseudo-gradient of the objective function that only requires two measurements of the objective function without considering the dimension of the optimization issue. Such feature allows for a significant decrease in the cost of optimization, especially in optimization problems with a large dimension [109].

1.3.6.1 Monkey Algorithm for optimization

In the following sections, the main components of the algorithm, i.e., the representation of the solution, the initialization, the climb process, the watch-jump process, and the somersault process are presented, respectively. All the details are listed as follows.

1.3.6.1.1 Representation of the solution

As first, an integer denoted as *M* is set as the population size of the Monkey Algorithm. Then, for each *i* monkey, its position is stated by the vector $\boldsymbol{x}_i = \left(x_{i1}, x_{i2}, ..., x_{in}\right)$, and such position is employed to express a solution of the optimization problem itself. In other words, the position $\boldsymbol{x}_i$ and the decision vector $\boldsymbol{x} = \left(x_1, x_2, ..., x_n\right)$ possess the same form, $i = 1, 2, ..., M$.

1.3.6.1.2 Initialization

For starting, it is mandatory to initialize the position for each monkey. For instance, one can assume that a region which contains the potential optimal solutions can be determined in advance. And for obtaining this, it is advisable to design a region with a nice shape, for example, an *n*-dimensional hypercube. Then a point is generated randomly from the hypercube. Such point is considered as the monkey position provided that it is feasible. Otherwise, one can sample points from the hypercube until a feasible point is found. So, repeating the process *M* times, *M* feasible points, $\boldsymbol{x}_i = \left(x_{i1}, x_{i2}, ..., x_{in}\right)$, are achieved and they are representative of the initial positions of monkeys $i = 1, 2, ..., M$, respectively.

1.3.6.1.3 Climb process

An enhancing to the objective function can be performed by the climb process. This process consists in a step-by-step procedure to change the initial positions of the monkeys to new ones. The gradient-based algorithm, such as the Newton's method, assume that information is available on the gradient vector associated with the objective function.
Anyway, there has been a growing interest in recursive optimization algorithms such as simultaneous perturbation stochastic approximation (SPSA) that do not depend on direct gradient information or measurements [109], [110]. These algorithms are based on the approximation to the gradient information to the gradient values of the objective function. The idea of SPSA [109] and the design of the climb process were exploited:

(1) randomly generate a vector $\Delta \boldsymbol{x}_i = \left(\Delta x_{i1}, \Delta x_{i2}, ..., \Delta x_{in}\right)$, where $j = 1, 2, ..., n$. The parameter a ($a > 0$), called the step length of the climb process, can be determined by specific situations

$$\Delta x_{ij} = \begin{cases} a, & \text{with probability } \dfrac{1}{2} \\ -a, & \text{with probability } \dfrac{1}{2} \end{cases} \qquad (27)$$

(2) calculate

$$f_{ij}^{'}\left(\boldsymbol{x}_i\right) = \frac{f\left(\boldsymbol{x}_i + \Delta \boldsymbol{x}_i\right) - f\left(\boldsymbol{x}_i + \Delta \boldsymbol{x}_i\right)}{2\Delta x_{ij}} \qquad (28)$$

where $j = 1, 2, ..., n$. The vector $f_i^{'}\left(\boldsymbol{x}_i\right) = \left(f_{i1}^{'}\left(\boldsymbol{x}_i\right), f_{i2}^{'}\left(\boldsymbol{x}_i\right), ..., f_{in}^{'}\left(\boldsymbol{x}_i\right)\right)$ is the pseudo-gradient of the objective function $f\left(\cdot\right)$ at the point $\boldsymbol{x}_i$.

(3) set $y_j = x_{ij} + a \cdot \text{sign}\left(f_{ij}^{'}\left(\boldsymbol{x}_i\right)\right)$, $j = 1, 2, ..., n$, and let $\boldsymbol{y} = \left(y_1, y_2, ..., y_n\right)$.

(4) update $\boldsymbol{x}_i$ with $\boldsymbol{y}$ provided that $\boldsymbol{y}$ is feasible. Otherwise, $\boldsymbol{x}_i$ is kept unchanged.

(5) repeat steps from (1) to (4) until there is a little change on the values of the objective function in the neighborhood iterations or the maximum allowable number of iterations has been reached.

1.3.6.1.4 Watch-Jump process

The climb process is followed by the watch-jump one. Indeed, each monkey arrived at its own mountaintop. Then, it takes a look and determine whether there are other points around it being higher than the current one. When the answer is yes, it jumps there from its current position. The maximal distance that a monkey can watch is defined as a positive parameter *b*. Now, the process performed by a monkey *i*, $i = 1, 2, ..., M$ can be illustrated:

(1) randomly generate real numbers y_j from $\left(x_{ij} - b, x_{ij} + b\right)$, $j = 1, 2, ..., n$. Let $\boldsymbol{y} = \left(y_1, y_2, ..., y_n\right)$.

(2) update $\boldsymbol{x}_i$ with $\boldsymbol{y}$ provided that both $f\left(\boldsymbol{y}\right) \geq f\left(\boldsymbol{x}_i\right)$ and $\boldsymbol{y}$ is feasible. Otherwise, repeat the step (1) until an appropriated $\boldsymbol{y}$ is found. One only replace $\boldsymbol{x}_i$ with that whose function value is greater than or equal to $f_{ij}^{'}\left(\boldsymbol{x}_i\right)$.

(3) repeat the climb process by employing $\boldsymbol{y}$ as an initial position.

1.3.6.1.5 The somersault process

The main task of the somersault process is to permits monkeys to find out new searching domains. The barycenter of current positions of each monkey has set as a pivot. Then, monkeys will somersault along the direction pointing to the pivot. In a more specific way, the monkey *i* will somersault to the next point from its current position $\boldsymbol{x}_i$ in the following way, $i = 1, 2, ..., M$:

(1) randomly generate a real number α from the interval $\left[c, d\right]$ (also called the somersault interval), where the somersault interval $\left[c, d\right]$ can be determined by specific situations.

(2) set

$$y_j = x_{ij} + \alpha\left(p_j - x_{ij}\right) \qquad (29)$$

where $p_j = \frac{1}{M}\sum_{i=1}^{M} x_{ij}$, $j = 1, 2, ..., n$. The point $\boldsymbol{p} = \left(p_1, p_2, ..., p_n\right)$ is the somersault pivot

(3) set $\boldsymbol{x}_i = \boldsymbol{y}$ if $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ is feasible. Otherwise, repeat steps (1) and (2) until a feasible solution $\boldsymbol{y}$ is found.

1.3.6.1.6 Termination

Following the climb process, the watch-jump process, and the somersault process, monkeys are ready for their next action. Indeed, the MA terminates after a fixed number of cyclic repetitions of the above described steps. In Figure 1.14, a pseudo-code of the MA is illustrated.

Monkey Search

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, ..., x_n)^T$

Generate initial population of M monkey in position i $\boldsymbol{x}_i = (x_{i1}, ..., x_{in})$

while *(criterion)*

  for *i=1:M*

    for *j=1:n (n-dimension of the hypercube)*

    Generate a vector and update the position $\boldsymbol{x}_i$ *and reach the climb number N,*

    Generate a real number and update the position $\boldsymbol{x}_i$ *and stop watch-jump process*

    Repeat the climb process

    Generate a real number in the interval $[c,d]$ *and update the position* $\boldsymbol{x}_i$

    Set the somersault pivot point

      and new ones/solutions are built/generated

    end for

  end for

  Rank the solutions and find the current best

end while

Display the optimal solution and objective value

Figure 1.14 Pseudo-code of Monkey Algorithm

A well-known issue is that the best position does not necessarily appear the last iteration, so the best one should be taken into account from the beginning. If the monkeys find a better one in the new iteration, the old one is replaced by it. Such position is reported as an optimal solution in the end of iterations.

1.3.6.1.7 Some applications

The Monkey Algorithm is one of the latest metaheuristic tools invented within the field of optimization. This algorithm has been employed in different fields, such as the network transmission [111], the control of frequency oscillations [112], the dynamic adaptation [113], for the optimization of power systems [114], and for sensors placement [115], [116].

1.4 References

[1] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms, 2nd Edition*, Luniver Press, 2010.

[2] K.E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, 1989.

[3] T. Weise, M. Zapf, R. Chiong, A.J. Nebro, *Why is optimization difficult?*, Studies in Computational Intelligence, 193, 1-50, 2009.

[4] S. Casciati, L. Elia, *Potential of two metaheuristic optimization tools for damage localization in civil structure*, Journal of Aerospace Engineering, 2015, submitted.

[5] R. Martí, G. Reinelt, *The Linear Ordering Problem, Exact and Heuristic methods in Combinatorial Optimization*, Springer, 2011.

[6] D. Coley, *An Introduction to Genetic Algorithms for Scientists and Engineers*, World Scientific Press, 1999.

[7] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1975.

[8] K.A. de Jong, Genetic Algorithms are NOT Function Optimizers, San Mateo, CA, USA, 1992.

[9] Y.W. Chen, Z. Nakao, K. Arakaki, S. Tamura, *Blind deconvolution based on genetic algorithms*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E80A(12), 2603-2607, 1997.

[10] T. Kawaguchi, T. Baba, R. Nagata, *3-D object recognition using a genetic algorithm-based search scheme*, IEICE Transactions on Information and Systems, E80D(11), 1064-1073, 1997.

[11] D.L. Carroll, *Chemical Laser Modeling with Genetic Algorithms*, AIAA Journal, 34(2), 338-346, 1996.

[12] D.L. Carroll, *Genetic Algorithm and Optimizing Chemical Oxygen-Iodine Lasers*, Developments in Theoretical and Applied Mechanics, Vol. XVII, Ed. Wilson, 1996.

[13] D. Halhal, G.A. Walters, D. Ouazar, D.A. Savic, *Water network rehabilitation with structured messy genetic algorithm*, Journal of Water Resources Planning and Management, ASCE, 123(3), 137-146, 1997.

[14] D.A. Savic, W.A. Walters, *Genetic Algorithms for least-cost design of water distribution networks*, Journal of Water Resources Planning and Management, ASCE, 123(2), 67-71, 1997.

[15] S. Migowsky, *Optimization of the energy consumption of a building using a genetic algorithm*, University of Exeter, thesis, 1995.

[16] H. Furuta, H. Hase, E. Watanabe, T. Tonegawa, H. Morimoto, *Application of the genetic algorithm to aesthetic design of dam structures*, Proceed. Neural Networks and Combinatorial Optimization in Civil and Structural Engineering Conference, Edinburgh, Scotland, 1993.

[17] C. Papadimitriou, J.L. Beck, S.-K. Au, *Entropy-based optimal sensor location for structural model updating*, Journal of Vibration and Control, 6(5), 781-800, 2000.

[18] L. Magnier, F. Haghighat, *Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network*, Building and Environment, 45(3), 739-745, 2010.

[19] C. Mares, C. Surace, *An application of genetic algorithms to identify damage in elastic structures*, Journal of Sound and Vibration, 195(2), 192-215, 1996.

[20] M.I. Friswell, J.E.T. Penny, S.D. Garvey, *A combined genetic and eigensensitivity algorithm for the location of damage in structures*, Computers and Structures, 69(5), 547-566, 1998.

[21] J.-H. Chou, J. Ghaboussi, *Genetic algorithm in structural damage detection*, Computers and Structures, 79(14), 1335-1353, 2001.

[22] S. Casciati, *Stiffness identification and damage localization via differential evolution algorithms*, Structural Control and Health Monitoring, 15(3), 436-449, 2008.

[23] E. Bonabeu, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

[24] M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italy, 1992.

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

[26] M. Dorigo, G. Di Caro, L.M. Gambardella, *Ant algorithms for discrete optimization*, Artificial Life, 5(2), 137-172, 1999.

[27] K.-S. Yoo, S.-Y. Han, *A modified ant colony optimization algorithm for dynamic topology optimization*, Computer and Structures, 123, 68-78, 2013.

[28] A. Farshidianfar, S. Soheili, *Ant colony optimization of tuned mass dampers for earthquake oscillations of high-rise structures including soil-structure interaction*, Soil Dynamic and Earthquake Engineering, 51, 14-22, 2013.

[29] L.L. Jiang, D.L. Maskell, J.C. Patra, *A novel ant colony optimization-based maximum power point tracking for photovoltaic systems under partially shaded conditions*, Energy and Buildings, 58, 227-236, 2013.

[30] A. Majumdar, A. De, D. Maity, D.K. Maiti, *Damage assessment of beams from changes in natural frequencies using ant colony optimization*, Structural Engineering and Mechanics, 45(2), 387-406, 2013.

[31] SA. Majumdar, B. Nanda, D. Maity, D.K. Maiti, *Structural damage detection based on modal parameters using continuous ant colony optimization*, Advances in Civil Engineering, 2014, 2014.

[32] A. Kaveh, B. Farahmand Azar, A. Hadidi, F. Rezazadeh Sorochi, S. Talatahari, *Performance-based seismic design of steel frames using ant colony optimization*, Journal of Constructional Steel Research, 66(4), 566-574, 2010.

[33] M. Serra, P. Venini, *On some applications of ant colony optimization metaheuristic to plane truss optimization*, Structural and Multidisciplinary Optimization, 32(6), 499-506, 2006.

[34] M. Dorigo, V. Maniezzo, A. Colorni, *Ant system: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26(1), 29-41, 1996.

[35] M. Dorigo, L.M. Gambardella, *Ant colony system: A cooperative learning approach to the traveling salesman problem*, IEEE Transactions on Evolutionary Computation, 1(1), 53-66, 1997.

[36] M. Dorigo, L.M. Gambardella, *Ant colonies for the travelling salesman problem*, BioSystems, 43(2), 73-81, 1997.

[37] R. Gan, Q. Guo, H. Chang, Y. Yi, *Improved ant colony optimization algorithm for the traveling salesman problems*, Journal of Systems Engineering and Electronics, 21(2), 329-333, 2010.

[38] J.B. Escario, J.F. Jimenez, J.M. Giron-Sierra, *Ant colony extended: Experiments on the travelling salesman problem*, Expert Systems with Applications, 42(1), 390-410, 2015.

[39] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, *Particle swarm and ant colony algorithms hybridized for improved continuous optimization*, Applied Mathematics and Computation, 188(1), 129-142, 2007.

[40] M.R. Jalali, A. Afshar, M.A. Mariño, *Multi-colony ant algorithm for continuous multi-reservoir operation optimization problem*, Water Resources Management, 21(9), 1429-1447, 2007.

[41] X.-S. Yang, J.M. Lees, C.T. Morley, *Application of virtual ant algorithms in the optimization of CFRP shear strengthened precracked structures*, Lecture Note in Computer Sciences, 3991, 834-837, 2006.

[42] F.C. Ergin, A. Yayimli, A.S. Uyar, *Survivable cross-layer virtual topology design using a hyper-heuristic approach*, Proc.,13th International Conference on Transparent Optical Networks, ICTON 2011; Stockholm, Sweden, June 2011.

[43] R. Bao, H. Pan, Q. Dong, L. Shao, *Ant colony-based routing algorithm for wireless sensor networks*, Chinese Journal of Sensors and Actuators, 24(11), 1644-1648, 2011.

[44] J. Kennedy, R.C. Eberhart, *Particle Swarm Optimization*, Proc., IEEE International Conference on Neural Networks, Piscataway, NJ, USA, 1942-1948, 1995.

[45] C. Elegbede , *Structural reliability assessment based on particles swarm optimization*, Structural Safety, 27(2), 171-186, 2005.

[46] A. Kaveh, S. Talatahari, *Size optimization of space trusses using Big Bang-Big Crunch algorithm*, Computers and Structures, 87(17-18), 1129-1140, 2009.

[47] Z.W. Geem, *Particle-swarm harmony search for water network design*, Engineering Optimization, 41(4), 297-311, 2009.

[48] G. Sun, G. Li, S. Hou, S. Zhou, W. Li, Q. Li, *Crashworthiness design for functionally graded foam-filled thin-walled structures*, Material Science and Engineering A, 527(7-8), 1911-1919, 2010.

[49] J. Chen, Y. Tang, R. Ge, Q. An, X. Guo, *Reliability design optimization of composite structures based on PSO together with FEA*, Chinese Journal of Aeronautics, 26(2), 343-349, 2013.

[50] J. Kennedy, R.C. Eberhart, Y. Shi, *Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation)*, Morgan Kaufmann Publ. Inc., 2001.

[51] J. Kennedy, *The behavior of particles*, Proc., 7th Annual Conference on Evolutionary Programming, Berlin, Germany, 1998.

[52] I.H. Osman, G. Laporte, *Metaheuristics: A Bibliography*, Annals on Operational Research, 63, 513-628, 1996.

[53] G. Laporte, I.H. Osman, *Metaheuristics in Combinatorial Optimizations, Annals on Operational Research*, 63, Baltzer Science Publishers, Basel, 1996.

[54] X.-S. Yang, *Harmony Search as a Metaheuristic Algorithm*, in: Music-Inspired Harmony Search Algorithm: Theory and Applications Studies in Computational Intelligence, Springer Berlin, 191, 1-14, 2009.

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

[56] D.G. Yoo, J.H. Kim, Z.W. Geem, *Overview of Harmony Search algorithm and its applications in Civil Engineering*, Evolutionary Intelligence, 7(1), 3-16, 2014.

[57] K.S. Lee and Z.W. Geem, *A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice*, Comput. Methods Appl. Mech. Engrg., 194, 3902-3933, 2005.

[58] Z.W. Geem, *Optimal cost design of water distribution networks using harmony search*, Engineering Optimization, 38, 259-280, 2006.

[59] Z.W. Geem, J.H. Jim, G.V. Loganathan, *Harmony search optimization: Application to pipe network design*, Intl. Journal of Modelling and Simulation, 22(2), 125-133, 2002.

[60] D.G. Yoo, S.M. Lee, J.H. Kim, *Optimal design of D-town network using multi-objective harmony search algorithm*, 14th Water Distribution Systems Analysis Conference 2012, WDSA 2012, 1, 350-367, 2012.

[61] D.G. Yoo, S.M. Lee, H. Jun, J.H. Kim, *A practice in optimal cost design of water distribution networks*, 14th Water Distribution Systems Analysis Conference 2012, WDSA 2012, 1, 642-651, 2012.

[62] Q. Luo, J. Wu, Y. Yang, J. Quian, J. Wu, *Optimal design of groundwater remediation system using a probabilistic multi-objective fast harmony search algorithm under uncertainty*, Journal of Hydrology, Part D, 3305-3315, 2014.

[63] G.F. de Medeiros, M. Kripka, *Optimization of reinforced concrete columns according to different environmental impact assessment parameters*, Engineering Structures, 59, 185-194, 2014.

[64] A. Akin, M.P. Saka, *Harmony search algorithm based optimum detailed design of reinforced concrete plane frames subject to ACI 318-05 provisions*, Computers and Structures, 147, 79-95, 2015.

[65] M.R. Maheri, M.M. Narimani, *An enhanced harmony search algorithm for optimum design of side sway steel frames*, Computer and Structures, 136, 78-89, 2014.

[66] P. Murren, K. Khandelwal, *Design-driven harmony search (DDHS) in steel frame optimization*, Engineering Structures, 59, 798-808, 2014.

[67] S. Nakrani and C. Tovey, *On Honey Bees And Dynamic Server Allocation in Internet Hosting Centers*, Adaptive Behavior, 12, 223-240, 2004.

[68] X.-S. Yang, *Engineering optimizations via nature-inspired virtual bee algorithms*, Proc., of IWINAC 2005, Lecture Notes In Computer Science, 3562, 317-323, 2005.

[69] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report, Erciyes University, 2005.

[70] B. Basturk, D. Karaboga, *An artificial bee colony (ABC) algorithm for numerical function optimization*, in: IEEE Swarm Intelligence Symposium 2006, Indianapolis, IN, USA, 2006.

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

[72] O.B. Haddad, A. Afshar, M.A. Mariño, *Honey bees mating optimization algorithm (HBMO); a new heuristic approach for engineering optimization*, Proc., the First International Conference on Modeling, Simulation and Applied Optimization (ICMSA0/05), Sharjah, UAE, 2005.

[73] O.B. Haddad, A. Afshar, M.A. Mariño, *Honey-bees mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization*, Water Resources Management, 20(5), 661-680, 2006.

[74] T. Niknam , A. Kavousifard, S. Tabatabaei, J. Aghaei, *Optimal operation management of fuel cell/wind/photovoltaic power sources connected to distribution networks*, Journal of Power Sources, 196(20), 8881-8896, 2011.

[75] D. Teodorović, *Swarm intelligence systems for transportation engineering: Principles and applications*, Transportation Research Part C: Emerging Technologies, 16(6), 651-667, 2008.

[76] M. Sonmez, *Discrete optimum design of truss structures using artificial bee colony algorithm*, Structural and Multidisciplinary Optimization, 43(1), 85-97, 2011.

[77] J.-Y. Park, S.-Y. Han, *Topology optimization for nonlinear structural problems based on artificial bee colony algorithm*, Intl. Jour. of Precision Engineering and Manufacturing, 16(1), 91-97, 2015.

[78] H. Sun, H. Waisman, R. Betti, *A multiscale flaw detection algorithm based on XFEM*, Intl. Jour. for Numerical Methods in Engineering, 100(7), 477-503, 2014.

[79] H. Sun, R. Betti, *Simultaneous identification of structural parameters and dynamic input with incomplete output-only measurements*, Structural Control and Health Monitoring, 21(6), 868-889, 2014.

[80] U. Topal, H.T. Öztürk, *Buckling load optimization of laminated plates via artificial bee colony algorithm*, 52(4), 755-765, 2014.

[81] X.-S. Yang, *Multiobjective firefly algorithm for continuous optimization*, Engineering with Computers, 29, 175-184, 2013.

[82] S. Casciati, L. Elia, *Potential of Metaheuristic Methods for Damage Localization and Stiffness Identification*, Proc., OPT-i International Conference on Engineering and Applied Sciences Optimization, Kos, Greece, 2014.

[83] S. Casciati, L. Elia, *Innovative, Bio-inspired, Metaheuristic Methods Targeted to SHM Diagnostic Goals*, Proc., 7SHMII – Seventh International Conference on Structural Health Monitoring of Intelligent Structures, Turin, Italy, 2015.

[84] L.F.F. Miguel, L.F. Fadel Miguel, R.H. Lopez, *A firefly algorithm for the design of force and placement of friction dampers for control of man-induced vibrations in footbridges*, Optimization and Engineering, 2014.

[85] G.-D. Zhou, T.-H. Yi, H.-N. Li, *Sensor placement optimization in structural health monitoring using cluster-in-cluster firefly algorithm*, Advances in Structural Engineering, 17(8), 1103-1115, 2014.

[86] F. Casciati, L. Elia, L. Faravelli, *Optimization of Sensors Location for Structural Monitoring*, Proc., OPT-i International Conference on Engineering and Applied Sciences Optimization, Kos, Greece, 2014.

[87] F. Casciati, L. Elia, L. Faravelli, *Optimization of Sensors Location and Actuator Control Law*, Proc., 6WCSCM – The Sixth World Conference on Structural Control and Monitoring, Barcelona, Spain, 2014.

[88] S. Talatahari, A.H. Gandomi, G.J. Yun, *Optimum design of tower structures using Firefly Algorithm*, Structural Design of Tall and Special Buildings, 23(5), 350-361, 2014.

[89] J.D. Altringham, *Bats: Biology and Behaviour*, Oxford University Press, 1996.

[90] X.-S. Yang, *A new metaheuristic bat-inspired algorithm*, in: Nature Inspired Cooperative Strategies for Optimization, Ed. Spinger, 284, 65-74, 2010.

[91] P. Richardson, *Bats*, Natural History Museum, London, 2008.

[92] X.-S. Yang, A.H. Gandomi, *Bat algorithm: A novel approach for global engineering optimization*, Engineering Computation, 29(5), 464-483, 2012.

[93] A.H. Gandomi, X.-S. Yang, A.H. Alavi, S. Talatahari, *Bat algorithm for constrained optimization tasks*, Neural Computing and Applications, 22(6), 1239-1255, 2013.

[94] O. Hasançebi, T. Teke, O. Pekcan, *A bat-inspired algorithm for structural optimization*, Computers and Structures, 128, 77-90, 2013.

[95] O. Hasançebi, S. Carbas, *Bat inspired algorithm for discrete size optimization of steel frames*, Advances in Engineering Software, 67, 173-185, 2014.

[96] R.B. Payne, M.D. Sorenson, K. Klitz, *The Cuckoos*, Oxford University Press, 2005.

[97] M.F. Shlesinger, *Search research*, Nature, 443, 282-282, 2006.

[98] X.-S. Yang, S. Deb, *Cuckoo Search via Lévy flights*, Proc., World Congress on Nature & Biologically Inspired Computing, NaBic 2009, IEEE Publications, USA, 210-214, 2009.

[99] X.-S. Yang, S. Deb, *Engineering optimisation by cuckoo search*, Intl. Jour. of Mathematical Modelling and Numerical Optimisation, 1(4), 330-343, 2010

[100] I. Pavlyukevich, *Cooling down Lévy flights*, J. Phys. A: Math. Theor., 40, 12299-12313, 2007.

[101] S. Guindon, O. Gascuel, *A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood*, Systematic Biology, 52(5), 696-704, 2003.

[102] A.H. Gandomi, X.-S. Yang, A.H. Alavi, *Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems*, Engineering with Computers, 29(1), 17-35, 2013.

[103] X.-S. Yang, S. Deb, *Multiobjective cuckoo search for design optimization*, Computers and Operations Research, 40(6), 1616-1624, 2013.

[104] A.H. Gandomi, S. Talatahari, X.-S. Yang, S. Deb, *Design optimization of truss structures using cuckoo search algorithm*, Structural Design of Tall and Special Buildings, 22(17), 1330-1349, 2013

[105] A. Kaveh, T. Bakhshpoori, M. Azimi, *Seismic optimal design of 3D steel frames using cuckoo search algorithm*, Structural Design of Tall and Special Buildings, 24(3), 210-227, 2015.

[106] M. Dhivya, M. Sundarambal, *Cuckoo Search for data gathering in Wireless Sensor Networks*, International Journal of Mobile Communications, 9(6), 642-656, 2011.

[107] M. Dhivya, M. Sundarambal, J.O. Vincent, *Energy efficient cluster formation in wireless sensor networks using Cuckoo Search*, Proc., 2nd International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2011, India, 2011.

[108] R. Zhao, W. Tang, *Monkey Algorithm for global numerical optimization*, Journal of Uncertain Systems, 2(3), 165-176, 2008.

[109] J. Spall, *An overview of the simultaneous perturbation method for efficient optimization*, John Hopkins APL Technical Digest, 19, 482-492, 1998.

[110] J. Kiefer, J. Wolfowitz, *Stochastic estimation of a regression function*, Ann. Math. Stat., 23, 462-466, 1952.

[111] J. Wang, Y. Yu, Y. Zeng, W. Luan, *Discrete monkey algorithm and its application in transmission*, in Proc., Power and Energy Society General Meeting, 2010 IEEE, Minneapolis, MN, USA, 2010.

[112] M. Aghababaei, M.M. Farsangi, *Coordinated control of low frequency oscillations using improved monkey algorithm*, Intl. Journal on Technical and Physical Problems of Engineering, 11(4), 13-17, 2012.

[113] L. Zheng, *An improved monkey algorithm with dynamic adaptation*, Applied Mathematics and Computation, 222, 645-657, 2013.

[114] C.M. Ituarte-Villarreal, N. Lopez, J.F. Espiritu, *Using the monkey algorithm for hybrid power systems optimization*, Procedia Computer Science, 12 344-349, 2012.

[115] N.J. Navimipour, S.H. Shabestari, V.S. Samaei, *Minimize energy consumption and improve the lifetime of heterogeneous wireless sensor networks by using monkey search algorithm*, Proc., Intl. Conference on Information and Knowledge Management, Singapore, 2012.

[116] T.-H. Yi, H.-N. Li, X.-D- Zhang, *Sensor placement on Canton Tower for health monitoring using asynchronous-climb monkey algorithm*, Smart Material and Structures, 21(12), 1-12, 2012.

Chapter 2 Selected tools and their development for structural optimization

In this chapter, a brief overview on the tools developed for structural optimization is outlined. Different source codes are taken as structure, and they are totally edited in order to frame structural control issues pursued during this research period.

2.1 The adopted algorithms

In this section, the adopted algorithms for carrying out the process of optimization are outlined and they are compared in terms of parameters and computational burden.
Since dozen metaheuristic methods are present in literature, the choice relapsed on those algorithms considered more innovative and more efficient in terms of accuracy of convergence, of versatility of the problem and computational time among the others.

2.1.1 Artificial Bee Colony Algorithm

In social insect colonies, each individual seems to behave with a detailed schedule. If an observer watch out for the group as a whole, it appears to be highly organized. For this reason among the others, the algorithms based on swarm intelligence and social insects begin to demonstrate their effectiveness and efficiency to solve difficult problems [1], [2], [3], [4], [5].
A swarm is defined as a group of multi-agent system such as bees. An important and interesting behavior of bee colonies is their foraging behavior, and in particular, how bees find a food source based on the amount of nectar and successfully bring nectar back to the hive. In a real bee colony, bees are divided as scout bees, employed bees and onlookers, as mentioned in Section 1.3.2.2.3.

In the ABC algorithm, the scout bees control the exploration process, while the employed bees and onlookers' carry out the exploitation process in the search space [6], [7].
So the population is equal to the number of both employed and onlooker bees. The employed bee turn into scout bees when their food source has been exhausted. A possible solution to the optimization problem becomes the position of an enhanced nectar amount of a food.
Given an objective function, few parameters have to be set for calibrating the algorithm. Such the set is composed by the number of iterations to be carried out during the process of optimization, the number of population that usually depends on the complexity of the issue, and a limit number, useful to abandon a food search if reached.
In literature, this algorithm has been tested and compared with other ones. For instance, the ABC is used for solving the multi-objective flexible job shop scheduling problem, and the experimental results on several well-known benchmarks show that is competitive to other recently published algorithms as the particle swarm optimization, the tabu search, the artificial immune algorithm [8]. Another example comes from the effect of the problem of dimensionality on the performance of a basic artificial bee colony, harmony search, and the bees algorithms on unimodal and multimodal well-known benchmark problems. A comparison of these algorithms was made in terms of the number of control parameters to be tuned. The algorithms considered were applied to proportional-integral-derivative (PID) controller tuning which involves the PID gains to be determined. Responses of the test systems to step input in terms of some metrics such as overshoot percentage, rising time, settling time and error are examined and also the effects of the disturbance on the control system performance and the system stability are analyzed [9].
In the field of composite structures, a generic method/model for multi-objective design optimization of laminated composite components is presented. The problem is formulated with multiple objectives of minimizing weight and the total cost of the composite component to achieve a specified strength. The primary optimization variables are the number of layers, its stacking sequence (the orientation of the layers) and thickness of each layer. The optimization method is validated for a number of different loading configurations and the performance is evaluated in comparison with other nature-inspired techniques which includes Particle Swarm Optimization (PSO), Artificial Immune System (AIS) and Genetic Algorithm (GA) [10].
For structural optimization, several contributions are proposed. For instance, truss weight is one of the most important factors in the cost of construction that should be reduced. Different methods have been proposed to optimize the weight of trusses. The artificial

bee colony algorithm is compared with the fly-back. The results indicate that the rate of convergence and the accuracy are optimized in comparison with other methods.
Another example falls on structural design optimization where a comparison of evolutionary-based optimization techniques is presented. In order to evaluate the proposed optimization approach a welded-beam design problem taken from the literature is solved. The proposed approach is applied to a welded-beam design problem and the optimal design of a vehicle component. The results show that the proposed approach gives better solutions compared to genetic algorithm, particle swarm, and immune algorithm [12].

2.1.2 Firefly Algorithm

The Firefly algorithm (FA), as described in Section 1.3.3, is based on the pattern of fireflies [13]. It is a recently developed swarm intelligence method, which has been inspired by the social behavior of fireflies, in which the brighter firefly attracts other darker fireflies. The stochastic, nature-inspired, and metaheuristic properties of the FA render it a promising tool for solving complex optimization problems. Generally, the firefly algorithm incorporates three strategies: attractiveness, distance between fireflies, and firefly movement.
This optimization tool is founded on five parameters that permit one to perform each requested issue. As in most of metaheuristic bio-inspired tools, the first parameter to be set is the population (which depends on the problem), the number of iterations, and three additional parameters: the attractiveness, whose value is between 0 and 1, the brightness, whose interval is the same of the previous one, and the absorption coefficient which is almost set equal to 1 (in most of optimization processes).
The FA is applied in several fields and it was used as comparing tool with other algorithms for demonstrating its strength in terms of convergence and computational burden. For instance, the extensive utilization of wireless sensor networks (WSNs) in SHM systems promotes optimal wireless sensor placement (OWSP) as an important topic. FA can be applied to the OWSP problem. The hybrid FA is applied to a long-span suspension bridge for verifications, and two other optimization methods, a simple discrete FA and a simple genetic algorithm, are also employed for facilitating comparisons. The results demonstrate that it can extract an optimal wireless sensor configuration with highly linear independence of identified mode shapes and outstanding WSN performance [14].
Such algorithm was also employed for simulating tensile loads in soil structures: the study developed an evolutionary metaheuristic intelligence model for efficiently and ac-

curately estimating reinforcement loads. The proposed model improves the prediction capability of the firefly algorithm (FA) by integrating intelligent components, namely, a chaotic map, an adaptive inertia weight, and a Lévy flight.
The method was then compared with conventional prediction methods in terms of the accuracy for predicting the reinforcement tensile loads of GRS structures. The cross-validation results demonstrated that the proposed model has a superior accuracy and mean absolute percentage errors lower than 10%. Moreover, a comparison with the baseline models and empirical methods indicate that the evolutionary metaheuristic intelligence model provides a significant improvement in terms of the root mean square errors (by 63.61–92.30%). This study validates the effectiveness of the proposed model for predicting reinforcement tensile loads and its feasibility for facilitating early designs of GRS structures [15].
Among the others, the damage localization is an important topic. Again, the FA is compared with other methods for validating its great performances: in structural health monitoring, the presence of damage is detected and localized by outlining the differences between the initial state and the current behavior of a given structure. The problem is often formulated as an optimization problem. Within a finite element discretization, some stiffness parameters are chosen as reference variables. Two metaheuristic tools, the artificial bee colony (ABC) algorithm and the firefly algorithm (FA), are applied to proceed the iterations toward the global minima of the objective function. By comparison between the identified and the analytical stiffness matrices, the damage detection and localization are performed. These methods are applied to a steel structure [16].

2.1.3 Cuckoo Search

A new and powerful tool is the Cuckoo Search (CS) inspired by some species of a bird family called cuckoo because of their special lifestyle and aggressive reproduction strategy. Such species lay their eggs in the nests of other host birds with amazing abilities such as selecting the recently spawned nests and removing existing eggs. On the other hand, some of the host birds are able to detect this parasite behavior of cuckoos get rid of the discovered alien eggs or even they build new nests in different locations.
CS usually outperforms better than many existing algorithms (e.g., GA and PSO) because there exists a fine balance of randomization and intensification and fewer control parameters have to be set [17]. They include the number of iterations, the population size, and the discovery rate, which is in the interval from 0 to 1. These few parameters make the CS less complex and thus more generic. Indeed, in the engineering field, several contribution proved the adaptability of the CS despite to other tools.

For instance, the Cuckoo Search (CS) algorithm for optimum tuning of PI controllers for Load Frequency Control (LFC) is suggested. Simulation results are introduced to show the enhanced performance of the CS based controllers in comparison with Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and conventional integral controller. These results denote that the proposed controllers offer better performance over others in terms of settling times and various indices [18]. Moreover, the Cuckoo Search (CS) method does not require repeated evaluation of fitness function and can provide a set of optimal solutions within a reasonable time if compared with the GA. Authors compare the application of GA and CS algorithm to the problem of design space exploration and conclude that CS perform better in terms of performance [19]. In the optimum design of two-dimensional steel frames the CS in combination with the Lévy flight was employed. The design algorithm is supposed to obtain minimum weight frame through suitable selection of sections from a standard set of steel sections such as the American Institute of Steel Construction (AISC) wide-flange (W) shapes. Strength constraints of AISC load and resistance factor design specification and displacement constraints are imposed on frames. In order to demonstrate the effectiveness and robustness of the CS, low-weight design and performance comparisons are made between the CS and other algorithms for some benchmark frames [20].
Focusing again on structural optimization, the CS has attracted much attention and wide applications, owing to its easy implementation and quick convergence. A hybrid cuckoo pattern search algorithm (HCPS) with feasibility-based rule is proposed for solving constrained numerical and engineering design optimization problems. This algorithm can combine the stochastic exploration of the cuckoo search algorithm and the exploitation capability of the pattern search method. Simulation and comparisons based on several well-known benchmark test functions and structural design optimization problems demonstrate the effectiveness, efficiency and robustness of the proposed HCPS algorithm despite to other algorithms as water cycle algorithm (WCA), cuckoo search, and swarm with intelligent information sharing (SIIS) [21].

2.1.4 Bat Algorithm

The bat algorithm (BA) is derived from echolocation behavior of bats. Echolocation is an advanced hearing based navigation system used by bats and some other animals to detect objects in their surroundings by emitting a sound to the environment. In general echolocation calls are characterized by three features: pulse frequency, pulse mission rate, and loudness (intensity).

The main parameters that have to be set are population size that usually varies in the interval from 10 to 25, the loudness sets from 0 to 1, and the pulse rate, which falls again in the interval from 0 to 1. It has been demonstrated that BA is much superior to other algorithms in terms of accuracy and efficiency [22].
Several challenges come from the comparison between the strength of BA despite to other algorithms. For instance, BA is examined in the context of discrete size optimization of steel frames designed for minimum weight. In the optimum design problem, frame members are selected from available set of steel sections for producing practically acceptable designs subject to strength and displacement provisions of American Institute of Steel Construction-Allowable Stress Design (AISC-ASD) specification. The performance of the technique is quantified using three real-size large steel frames under actual load and design considerations. The results obtained provide a sufficient evidence for successful performance of the BA in comparison to other metaheuristics employed in structural optimization, as the HS, the AHS, and the BB-BC among the others [23]. In other engineering field, as robotics, the BA is employed and its power is underlined by the good results. Indeed, robot has to move from its starting point called source point to final point called as destination point with minimum number of moves and iteration. Both cuckoo search and bat algorithms are applied for the proposed problem and simulation results are compared. The techniques are applied for different number of population and bat algorithm provide better results as compared to cuckoo search [24].

2.2 Experimentation with benchmark mathematical functions

After the description of the employed algorithms for the optimization process in structural monitoring, in this section some classical benchmarks are exploited for validating the algorithms performances [25].
Indeed, the main issue of the optimization is to achieve the best values of the variables of a function that has to be optimized. Despite to the deterministic algorithms, metaheuristic ones are not affected by the behavior of the optimization problem. For this reason, such algorithms are widely usable [26].
When one deals with evolutionary computation, it is common to compare different methods using a large test set, in particular whether the set involves the optimization of functions. Notwithstanding, the effectiveness of an algorithm cannot be weighted by the number of problems that it can solves better than another algorithm. For instance, if one is comparing two searching algorithms with a set of possible functions, the performance may be, on average, the same. Consequently, it is needed to improve an attempt to design

a *perfect* test set, where all the functions are present for determining whether an algorithm is better than another one for that function. Moreover, this is the reason why, during the evaluation of an algorithm, one has to look for the typology of problems where its performance is good, for identifying the type of problems for which an algorithm is more suitable. For carrying out such purpose, a previous study of the functions to be optimized for constructing a test set with four benchmark functions is performed. This allows one to draw conclusions of the performance of the algorithm depending on the type of function. The adopted functions are grouped in Table 2.1.

Table 2.1 Mathematical functions

Function	Expression
Sphere	$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[\alpha \left(x_i^2 - x_i^2 - 1 \right)^2 + \left(x_i - 1 \right)^2 \right]$ where $\alpha = 100$
Rastrigin	$f(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} \left(x^2 - 10\cos\left(2\pi x_i \right) \right)$
Griewank	$f(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$

The shape of each function is represented in Figure 2.1.

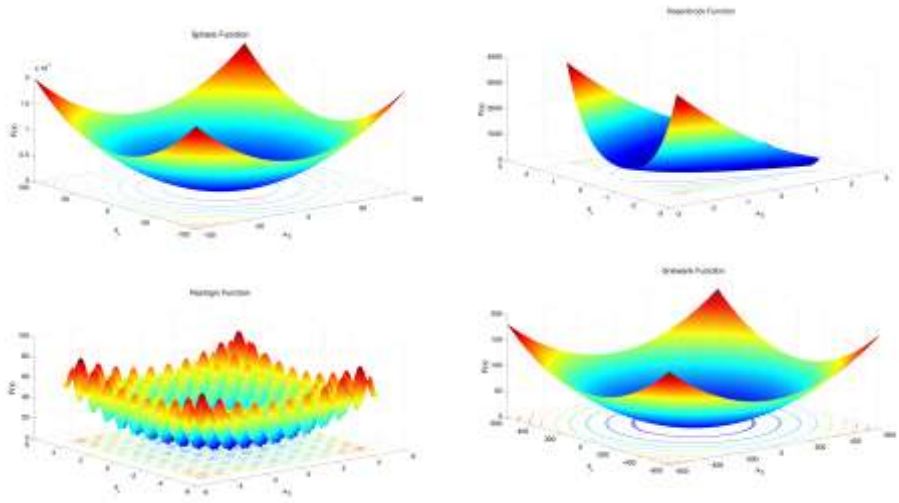


Figure 2.1 Benchmark functions

Herein, the compared algorithm are the Artificial Bee Colony (ABC), the Firefly Algorithm (FA), the Cuckoo Search (CS), and the Bat Algorithm (BA).
Each algorithm is based on a set of parameters, as shown in Table 2.2, and for better performing the comparison, the population and the number of iterations of the algorithms are kept constant for all the analyses carried out, with a value equal to 10 and 500, respectively.

Table 2.2 Comparison between the parameters of each algorithm

ABC	FA	CS	BA
# iterations	# iterations	# iterations	# iterations
population	population	nests	population
	attractiveness		loudness
limit	brightness	rate of alien egg	pulse rate
	absorption coefficient		

Each benchmark function, namely Sphere, Rosenbrock, Rastrigin, and Griewank, is applied as objective function of the selected algorithms. After one-hundred runs per algorithm, the mean and the standard deviation are presented in Table 2.3, and then some conclusions are outlined.

Table 2.3 Simulation results

Function	Algorithm							
	ABC		FA		CS		BA	
	mean	st. dev.	mean	st. dev.	mean	st. dev.	mean	st. dev.
Sphere	6.2×10^{-14}	6.2×10^{-13}	3.1×10^{-8}	2.4×10^{-8}	6.1×10^{-4}	1.3×10^{-3}	2.3×10^{-6}	2.8×10^{-6}
Rosenbrock	3.0×10^{-4}	2.8×10^{-3}	2.2×10^{-8}	5.1×10^{-8}	1.0×10^{-5}	0.9×10^{-4}	2.9×10^{-6}	3.1×10^{-6}
Rastrigin	8.3×10^{-1}	1.2×10^{-0}	3.2×10^{-7}	8.5×10^{-7}	3.7×10^{-11}	9.6×10^{-10}	3.2×10^{-6}	2.4×10^{-6}
Griewank	2.8×10^{-4}	2.5×10^{-3}	5.4×10^{-8}	3.3×10^{-7}	5.0×10^{-4}	7.0×10^{-4}	6.1×10^{-6}	6.0×10^{-5}

The behavior of the algorithms along the performed tests has been stable and always reach good results in terms of accuracy of the best solution, and also in the time employed for each analysis (the mean time for the ABC is about 1.20sec, for the FA is 0.35sec, for the

CS is 0.25sec, and for BA is 0.20sec). I can conclude that the FA and the CS are the most balanced in terms of accuracy and computational burden.

2.3 References

[1] R. Le Riche, R.T. Haftka, *Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm*, AIAA Journal 31(5), 1993.

[2] A.K. Garg, D. Roy Mahapatra, S. Suresh, S. Gopalakrishna, S.N. Omkar, *Estimation of composite design model parameters using spectral element and neural networks*, Composite Science and Technology 64(16), 2477-2494, 2007.

[3] S.N. Omkar, D. Mudigere, G. Narayana Naik, S. Gopalakrishna, *Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures*, Computers & Structures 86, 1-14, 2008.

[4] K.E. Parsopoulos, D.E. Tasoulis, M.N. Vrahatis, *Multi-objective optimization using parallel vector evaluated particle swarm optimization*, Proc., of International Conference on Artificial Intelligence and Applications (IASTED), 2004.

[5] S.N. Omkar, R. Khandelwal, S. Yathindra, G. Narayana Naik, S. Gopalakrishnan, *Artificial immune system for multi-objective design optimization of composite structures*, Engineering Applications of Artificial Intelligence 21(8),1416-1429, 2008.

[6] D. Karaboga, B. Basturk, *On the performance of Artificial Bee Colony (ABC) algorithm*, Applied Soft Computing 8 (1), 687-697, 2008.

[7] X.-S. Yang, *Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms*, Lecture Notes in Computer Science, vol. 3562, Springer-Verlag GmbH, 317-323, 2005.

[8] J.-Q. Li, Q.-K. Pan, K.-Z. Gao, *Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems*, J. Adv. Manuf. Technol., 55, 1159-1169, 2011.

[9] D. Karaboga, B. Akay, *Proportional-integral-derivative controller design by using artificial bee colony, harmony search, and the bees algorithms*, Proc., the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering, 224(7), 869-883, 2010.

[10] S.N. Omkar, J. Senthilnath, Rahul Khandelwal, G. Narayana Naik, S. Gopalakrishnan, *Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures*, Applied Soft Computing, 11, 489-499, 2011.

[11] A.R. Fiouz, M. Obeydi, H. Forouzani, A. Keshavarz, *Discrete optimization of trusses using an artificial bee colony (ABC) algorithm and the fly-back mechanism*, Structural Engineering and Mechanics, 44(4), 501-519, 2012.

[12] A.R. Yildiz, *Comparison of evolutionary-based optimization algorithms for structural design optimization*, Eng. Appl. of Artif. Intell., 26, 327-333, 2013.

[13] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms, 2nd Edition*, Luniver Press, 2010.

[14] G.-D. Zhou, T.-H. Yi, H. Zhang, H.-N. Li, *Energy-aware wireless sensor placement in structural health monitoring using hybrid discrete firefly algorithm*, Structural Control & Health Monitoring, 22, 648-666, 2015.

[15] J.-S. Chou, K.-H. Yang, J.P. Pampang, A.-D. Pham, *Evolutionary metaheuristic intelligence to simulate tensile loads in reinforcement for geosynthetic-reinforced soil structures*, Computers and Geotechnics, 66, 1-15, 2015.

[16] S. Casciati, L. Elia, *Potential of two metaheuristic optimization tools for damage localization in civil structures*, Journal of Aerospace Engineering, submitted, 2015.

[17] X.-S. Yang, S. Deb, *Engineering optimization by cuckoo search*, Int. J. Math. Modelling & Numerical Optimisation, 1, 330-343, 2010.

[18] A.Y. Abdelaziz, E.S. Ali, *Cuckoo Search algorithm based load frequency controller design for nonlinear interconnected power system*, Electrical Power and Energy Systems, 73, 632-643, 2015.

[19] A. Kumar, S. Chakarverty, *Design optimization using Genetic Algorithm and Cuckoo Search*, Proc., 2011 IEEE international Conference on Electro/information Technology, Mankato, MN, USA, May, 2011.

[20] A. Kaveh, T. Bakhshpoori, *Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights*, Struct. Design Tall Spec. Build., 22, 1023-1036, 2013.

[21] W. Long, W.-Z. Zhang, Y.-F. Huang, Y.-X. Chen, *A hybrid cuckoo search algorithm with feasibility-based rule for constrained structural optimization*, J. Cent. South Univ., 21, 3197-3204, 2014.

[22] X.-S. Yang, A.H. Gandomi, *Bat algorithm: a novel approach for global engineering optimization*, Engineering Computations, 29(5), 464-483, 2012.

[23] O. Hasançebi, S. Carbas, *Bat inspired algorithm for discrete size optimization of steel frames*, Advances in Engineering Software, 67, 173-185, 2014.

[24] Y.Gigras, K. Gupta, K. Choudhary, *A Comparison between Bat Algorithm and Cuckoo Search for Path Planning*, International Journal of Innovative Research in Computer and Communication Engineering, 3(5), 2015

[25] O. Castillo, P. Mellin, *Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics: Theory and Applications*, Studies in Computational Intelligence, 574, Springer, 2015.

[26] C. Solano-Aragón, O. Castillo, *Optimization of benchmark mathematical functions using the firefly algorithm with dynamic parameters*, Studies in Computational Intelligence, 574, Springer, 2015.

Chapter 3 Numerical modeling and examples

In this chapter, the numerical examples carried out during this research period are presented.
The structural models of these structures were designed within the theory of Finite Element Theory (FEM) [1]. The adopted software employed to perform the models are principally two: the first one is developed by the MSC® Software House, while the second by the MathWorks® Software House. In particular the graphical interfaces are called Marc Mentat [2] and MATLAB® [3]. The first software allows one a complete solution, i.e. from the pre-processing to the post-processing solutions, for the nonlinear finite element analysis (FEA), and the second one makes the user able to manage a large amount of data and perform specific analysis, as, for instance, the optimization process. Furthermore, Marc Mentat provides the option to model and simulate the response of a structure under different scenarios. From the analysis carried out by this software, one can extract the results that can be used into a Matlab code. This operation makes the user follow the process of optimization, or the model reduction steps, and so forth.

3.1 The formulation of the objective function

The dynamic signature of a structure is assumed as either experimentally measured or numerically simulated. In order to extract the modal features of a structure, for instance frequencies and mode shapes, traditional modal analysis tools for linear systems can be exploited. The obtained parameters which refer to the actual behavior of the structure in its current conditions are stated with the subscript '*exact*' in each following elaboration. Particularly, $\bar{\boldsymbol{\omega}}_{exact}$ is the $N \times 1$ vector of known natural frequencies, and $\bar{\boldsymbol{\Phi}}_{exact}$ is the $N \times N$ matrix of the corresponding modal shapes.

Let $\boldsymbol{x}$ be a generic $d\times 1$ vector of design parameters in the actual population along the firefly algorithm (FA). For easiness of notation, the superscript tracking the population and the subscript that identifies the individual are herein dropped. The corresponding stiffness matrix is stated as $\mathbf{K}_{gen}\left(\boldsymbol{x}\right)$, thus emphasizing that is assessed at the beginning of the current step of the FA.

Thus, both frequencies and mode shapes are evaluated by solving the following eigenvalues-eigenvectors problem:

$$\left[\mathbf{K}_{gen}\left(\boldsymbol{x}\right)-\omega_{i,gen}^{2}\left(\boldsymbol{x}\right)\mathbf{M}\right]\boldsymbol{\phi}_{i,gen}\left(\boldsymbol{x}\right)=\mathbf{0}\qquad i=1,...,N \quad (30)$$

where **M** states the mass matrix assumed as known and unvaried during with the initial state. Thus, the resulting eigenvalues are stored in a $N\times 1$ vector $\boldsymbol{\omega}_{gen}\left(\boldsymbol{x}\right)$, while the eigenvectors in a $N\times N$ matrix $\boldsymbol{\Phi}_{gen}\left(\boldsymbol{x}\right)$, where $\boldsymbol{\phi}_{i,gen}\left(\boldsymbol{x}\right)$ is the *i*-th column.

Hence, the objective function can be formulated as the norm of the difference between the exacted and the generated parameters, and in its matrix notation can be expressed as:

$$F\left(\boldsymbol{x}\right)=\sqrt{\left[\left(\frac{\bar{\boldsymbol{\omega}}_{exact}-\boldsymbol{\omega}_{gen}\left(\boldsymbol{x}\right)}{\bar{\boldsymbol{\omega}}_{exact}}\right)^{\mathrm{T}}\cdot\mathbf{P}\cdot\left(\frac{\bar{\boldsymbol{\omega}}_{exact}-\boldsymbol{\omega}_{gen}\left(\boldsymbol{x}\right)}{\bar{\boldsymbol{\omega}}_{exact}}\right)\right]}+wG\left(\bar{\boldsymbol{\Phi}}_{exact},\boldsymbol{\Phi}_{gen}\left(\boldsymbol{x}\right)\right) \quad (31)$$

or, equivalently, in its explicit scalar form as:

$$F\left(\boldsymbol{x}\right)=\sqrt{\sum_{i=1}^{N}\frac{1}{i}\left(\frac{\bar{\boldsymbol{\omega}}_{i,exact}-\boldsymbol{\omega}_{i,gen}\left(\boldsymbol{x}\right)}{\bar{\boldsymbol{\omega}}_{i,exact}}\right)^{2}}+w\max_{1\le j\le N}\left[\frac{\sum_{i=1}^{N}\left(\bar{\boldsymbol{\Phi}}_{exact}-\boldsymbol{\Phi}_{ij,gen}\left(\boldsymbol{x}\right)\right)^{2}}{\sum_{i=1}^{N}\boldsymbol{\Phi}^{2}{}_{ij,gen}}\right] \quad (32)$$

being **P** a $N\times N$ diagonal matrix of weights, where the *i*-th element ($1/i$) is chosen to prioritize the lower frequencies over the higher ones, surely affected by the measurements noise.

The first term in the right hand side of Equation (31) has been, initially, considered alone. The further introduction of the mode shapes results in an enhancement for the convergence of the method, as confirmed by the numerical results reported in the following sections. Then, a scalar weight, namely *w*, is introduced and preliminary calibrated by manually running the code several times in order to achieve satisfying results. It is worth noting that, when the eigenvectors are considered, the norm of a matrix is not uniquely defined. Indeed, according to [4], the norm of a $N\times N$ matrix **A** can be calculated as either:

$$l_2\text{norm}(\mathbf{A}) = \left(\text{greatest eigenvalue of } \mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{1/2} \quad (33)$$

or

$$l_\infty\text{norm}(\mathbf{A}) = \max_{1\le j\le N}\sum_{i=1}^{N}\left(A_{ij}\right) \quad (34)$$

When the $l_2$norm is applied to the j-th column of the matrix **A**, it degenerates to the traditional norm of a vector, namely $l_2\text{norm}(\mathbf{A}_\mathbf{j}) = \|\mathbf{A}_\mathbf{j}\| = \sqrt{\sum_{i=1,\ldots,N}\left(A_{ij}^2\right)}$ with $j = 1,\ldots,N$. The last term on the right hand side of Equation (31) is given as the maximum of the ratios between the squares of the $l_2$norm of the vectors $\left[\bar{\phi}_{j,exact} - \phi_{j,gen}(\boldsymbol{x})\right]$ and $\bar{\phi}_{j,exact}$, for $j = 1,\ldots,N$. In the mathematical form, one obtains:

$$G\left(\bar{\mathbf{\Phi}}_{exact}, \mathbf{\Phi}_{gen}(\boldsymbol{x})\right) = \max_{1\le j\le N}\frac{\left\|\bar{\phi}_{j,exact} - \phi_{j,gen}(\boldsymbol{x})\right\|^2}{\left\|\bar{\phi}_{j,exact}\right\|^2} \quad (35)$$

whose scalar expression is explicitly represented by the second term in Equation (32). Finally, the optimization problem can be formulated as follows:

$$\begin{aligned}&\text{minimize } F(\boldsymbol{x})\\ &\text{under the constraint: } \boldsymbol{x}_{Lb} \le \boldsymbol{x} \le \boldsymbol{x}_{Ub}\end{aligned} \quad (36)$$

being $\boldsymbol{x}_{Lb}$ and $\boldsymbol{x}_{Ub}$ the $d \times 1$ vectors containing the lower and the upper bounds of each variable in the design parameter vectors, respectively.

3.2 The cantilever beam

In this section, a first example of a cantilever beam is outlined [5]. The analyses were performed in the MATLAB® environment, including the finite element analyses for assuring the transparency at each step of the procedure. Figure 3.1 shows a short cantilever beam of length 10.16cm and height 5.06cm that is chosen as case study [6].

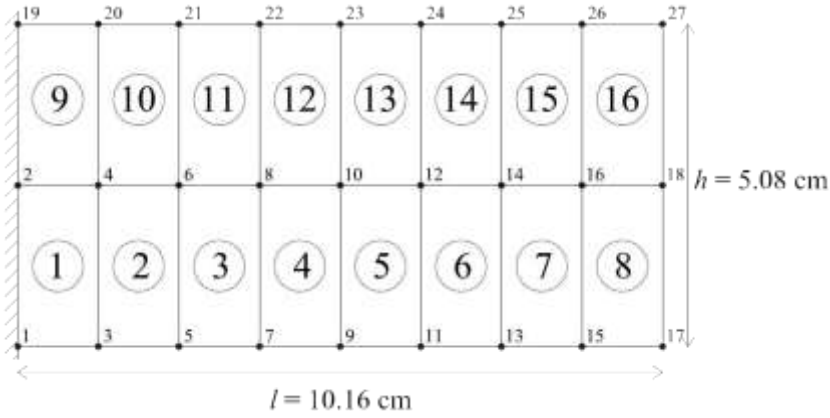


Figure 3.1 Finite element discretization of the cantilever beam

The beam is meshed in sixteen, two-dimensional, four-node, iso-parametric elements over two layers, under the assumption of plane stress condition. Hence, the discretization consists of 16 elements and 27 nodes with three of them fixed, so one can compute the number of degrees of freedom. The material is isotropic and each element has Young modulus, Poisson ratio, and a mass density. Table 3.1 summarizes all the features above described.

Table 3.1 Features of the cantilever beam

Property	Value
Height	5.08cm
Length	10.16cm
Number of nodes per element (n_e)	4
Number of elements (m)	16
Number of nodes (n)	27
Number of fixed nodes (n_f)	3
Number of degrees of freedom ($n_{dof} = 2 \cdot (n - n_f)$)	48
Young modulus (E)	703.7MPa
Poisson ratio (v)	0.3
Mass density (ρ)	$7.7 \times 10^{-5} \left(\mathrm{N/mm^3}\right) / \left(\mathrm{mm/s^2}\right)$

A classical displacement-based approach is adopt to carry out each finite element analysis [7]. In this manner, local displacements of a single element are expressed as functions of its nodal displacements by an approximate model that depends on the selection of the shape functions, namely $\mathbf{N}_e$.

In such context, the elementary stiffness matrix, $\mathbf{K}_e$, whose size is $2n_e \times 2n_e$, follows the well-known principle of virtual works and is expressed by

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}^{\mathrm{T}}{}_e \mathbf{D}_e \mathbf{B}_e dV \qquad (37)$$

where V_e denotes the volume of the *e*-th finite element, $\mathbf{B}_e$, the compatibility matrix containing the spatial derivatives of shape functions, and $\mathbf{D}_e$ the material constants.
Then the connectivity matrix, $\mathbf{L}_e$, has to be defined and thus one assemble the global stiffness matrix, whose size is $n_{dof} \times n_{dof}$

$$\mathbf{K} = \sum_{e=1}^{m} \mathbf{L}^{\mathrm{T}}{}_e \mathbf{K}_e \mathbf{L}_e \qquad (38)$$

According to this procedure, the continuity of the structure is imposed at each node in commons between different elements that must "follow" the same displacement passing from the local to the global reference system. Following this procedure, the displacements prevented by the boundary conditions are properly deleted during the process of assembling.
In a similar way, after the introduction of the material mass density ρ, the elementary mass matrix can be defined as

$$\mathbf{M}_e = \int_{V_e} \rho \mathbf{N}_e^{\mathrm{T}} \mathbf{N}_e dV \qquad (39)$$

and consequently, the assembled mass matrix, of size $n_{dof} \times n_{dof}$, can be given as

$$\mathbf{M} = \sum_{e=1}^{m} \mathbf{L}^{\mathrm{T}}{}_e \mathbf{M}_e \mathbf{L}_e \qquad (40)$$

After the completion of this process, the resulting stiffness and mass matrices are symmetric and positive-definite. So, the $1 \times n_{dof}$ vector of unknown displacements, namely $\mathbf{u}$, is governed by the classical equation of motion of an undamped system in free vibration, $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\dot{\mathbf{u}} = \mathbf{0}$.
In this example, as in others that will be treated in the next sections, the damage is introduced to any *e*-th element of the structure by simply pre-multiplying the corresponding local stiffness matrix by a dimensionless quantity, namely α_e, whose real values falls in the interval $[0,1]$. For sake of simplicity, a degradation of stiffness in the *e* element provides to a damaged stiffness matrix, expressed as

$$\mathbf{K}_{e,dam} = \alpha_e \mathbf{K}_e \qquad (41)$$

with $0 < \alpha_e < 1$, being $\alpha_e = 1$ associated to the undamaged reference condition. Each analysis carried out for different damage scenarios as summarized in Table 3.2 and in Figure 3.2.

Table 3.2 Different structural configurations and damage scenarios (cantilever beam)

Structural configuration	Damaged element(s)	α_e
Undamaged	-	1
Damage A	1	0.8
Damage B	9	0.8
Damage C	1 – 9	0.8
Damage D	1	0.4
Damage E	9	0.4
Damage F	12	0.8
Damage G	1 – 2 – 9 – 10	0.8 – 0.9 – 0.7 – 0.8

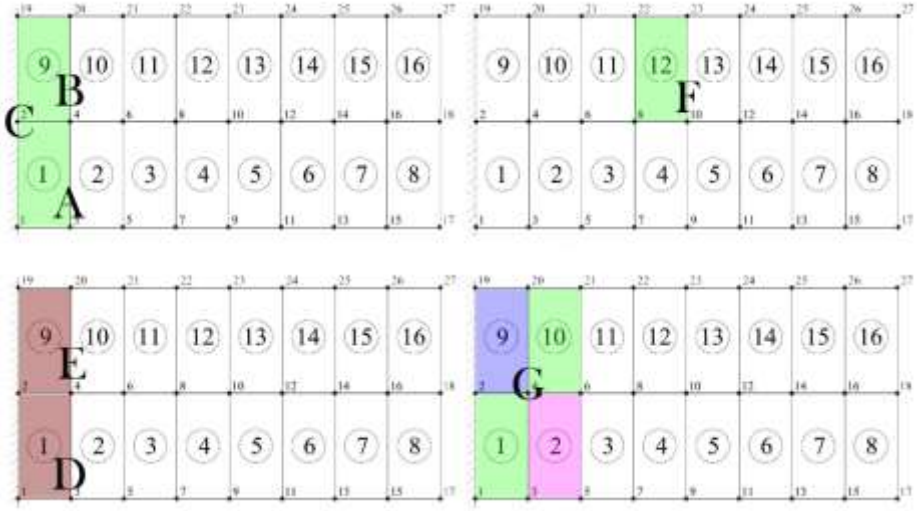


Figure 3.2 Different damage scenarios for each structural configuration (cantilever beam)

It is worth noting that the construction of the mass matrix can be faced using two different approaches:

(1) the herein adopted finite elements method, based on a consistent mass matrix (i.e., a full matrix of non-null inertia terms, which includes the rotational inertia) [8];

(2) the classical assumption of lumped masses at nodal points, which provides a diagonal mass matrix as $\mathbf{M}_{L_e} = (\rho V_e / n_e)\mathbf{I}_e$, where $\mathbf{I}_e$ is the identity matrix of size $2n_e \times 2n_e$.

After this assumption, the exact values of natural frequencies and mode shapes are achieved by solving the eigenproblem stated in Equation (30) for all the structural configurations proposed in Table 3.2.

Table 3.3 First exact nine modal frequencies for different values of element stiffness coefficient (cantilever beam)

Structural configuration	Exact values of first 9 circular frequencies (rad/s)								
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
Undamaged	66.86	235.38	262.48	562.97	701.09	733.97	956.56	965.88	1059.69
Damage A	65.41	232.10	258.81	555.75	692.53	727.73	955.39	959.60	1046.85
Damage B	65.41	232.10	258.81	555.75	692.53	727.73	955.39	959.60	1046.85
Damage C	63.98	228.67	254.64	548.06	684.12	720.89	953.07	954.65	1036.58
Damage D	60.03	220.53	248.33	533.26	660.24	710.43	941.83	952.27	999.05
Damage E	60.03	220.53	248.33	533.26	660.24	710.43	941.83	952.27	999.05
Damage F	66.44	233.17	260.35	557.90	699.04	733.11	952.10	955.97	1047.43
Damage G	61.73	222.33	248.70	540.96	674.26	713.24	944.25	953.27	1027.94

The resulting nine circular frequencies are reported in Table 3.3 for each considered case. In all the performed analyses, both eigenvalues and eigenvectors are employed as input parameters of the solving algorithm. Then, a unit weight, namely *w*, is chosen in the objective function of Equation (32) to account for the eigenvectors' contribution [9].
Once the modal features are identified, both frequencies and modes are introduced in the objective function and, hence the algorithm, namely ABC and FA, are applied in to deal with the optimization problem pinpointed in Equation (36). Each performed analyses aim to reach the damage scenarios stated in Table 3.2. The following $d \times 1$ (where $d = m$) parameters vector collects the coefficients of the stiffness matrix which have to be identified:

$$\boldsymbol{x} = \left[\alpha_1 \;\; \alpha_2 \;\; \ldots \;\; \alpha_n\right]^{\mathrm{T}} \qquad (42)$$

Hence, in order to center the adequate search domain, a preliminary study on the undamaged structure is performed. Such an operation is useful for facilitating the convergence by reducing the size of the search domain. Furthermore, the undamaged configuration is investigated to calibrate the control parameters of each algorithm.

In following subsections, the analyses carried out via artificial bee colony and firefly algorithm both on the undamaged and damaged structures are presented.

3.2.1 Preliminary analyses on undamaged structures

The control parameters of the employed metaheuristic tools are summarized from Table 3.4 to Table 3.7. For instance, for FA, parameters as the randomization number, the minimum attractiveness and the absorption coefficient, are maintained constant in all the performed analyses, as in most of the implementations found in literature [10]; the same procedure was adopted for the parameters of other algorithms.

Table 3.4 Control parameters of ABC (cantilever beam)

Control parameters of ABC	Values for each structural configuration						
	D_{ABC}-A	D_{ABC}-B	D_{ABC}-C	D_{ABC}-D	D_{ABC}-E	D_{ABC}-F	D_{ABC}-G
CS, size of the initial population of honeybees	32	32	32	32	32	32	32
C_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000	3000
λ, limit number for abandoning the food search	100	100	100	100	100	100	100

Table 3.5 Control parameters of FA (cantilever beam)

Control parameters of FA	Values for each structural configuration						
	D_{FA}-A	D_{FA}-B	D_{FA}-C	D_{FA}-D	D_{FA}-E	D_{FA}-F	D_{FA}-G
NP, size of the initial population of fireflies	30	30	30	30	30	30	30
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000	3000
ζ, randomization number	0.5	0.5	0.5	0.5	0.5	0.5	0.5
β_{min}, minimum attractiveness	0.2	0.2	0.2	0.2	0.2	0.2	0.2
γ, absorption coefficient	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 3.6 Control parameters of CS (cantilever beam)

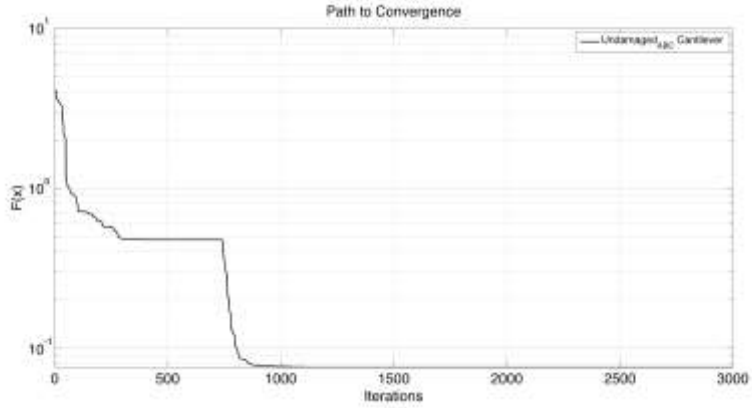
Control parameters of CS	Values for each structural configuration						
	D_{CS}-A	D_{CS}-B	D_{CS}-C	D_{CS}-D	D_{CS}-E	D_{CS}-F	D_{CS}-G
n, number of nests	40	40	40	40	40	40	40
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000	3000
p_a, discovery rate	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table 3.7 Control parameters of BA (cantilever beam)

Control parameters of BA	Values for each structural configuration						
	D_{CS}-A	**D_{CS}-B**	**D_{CS}-C**	**D_{CS}-D**	**D_{CS}-E**	**D_{CS}-F**	**D_{CS}-G**
n, population of bats	50	50	50	50	50	50	50
I_{max}, maximum number of iterations	1500	1500	1500	1500	1500	1500	1500
pulse rate	0.8	0.8	0.8	0.8	0.8	0.8	0.8
loudness	1	1	1	1	1	1	1

The search domain is centered around $x_0 = 1.0$; it is assigned as an interval, $x_{Lb} \le x_0 \le x_{Ub}$, with $x_{Lb} = 0.8$ and $x_{Ub} = 1.2$.

Under these assumptions, for the structural configuration labelled as *Undamaged Cantilever* the solution is reached with an error close to zero after 1096 iterations by ABC, after 947 by FA, after 2633 by CS, and after 795 by BA. Figure 3.3 shows the path to convergence for the undamaged cases. Moreover, a study on the computational burden is investigated and the duration of analyses are 7160 seconds for ABC, 480 seconds for FA, 701 for CS, and 377 for BA.



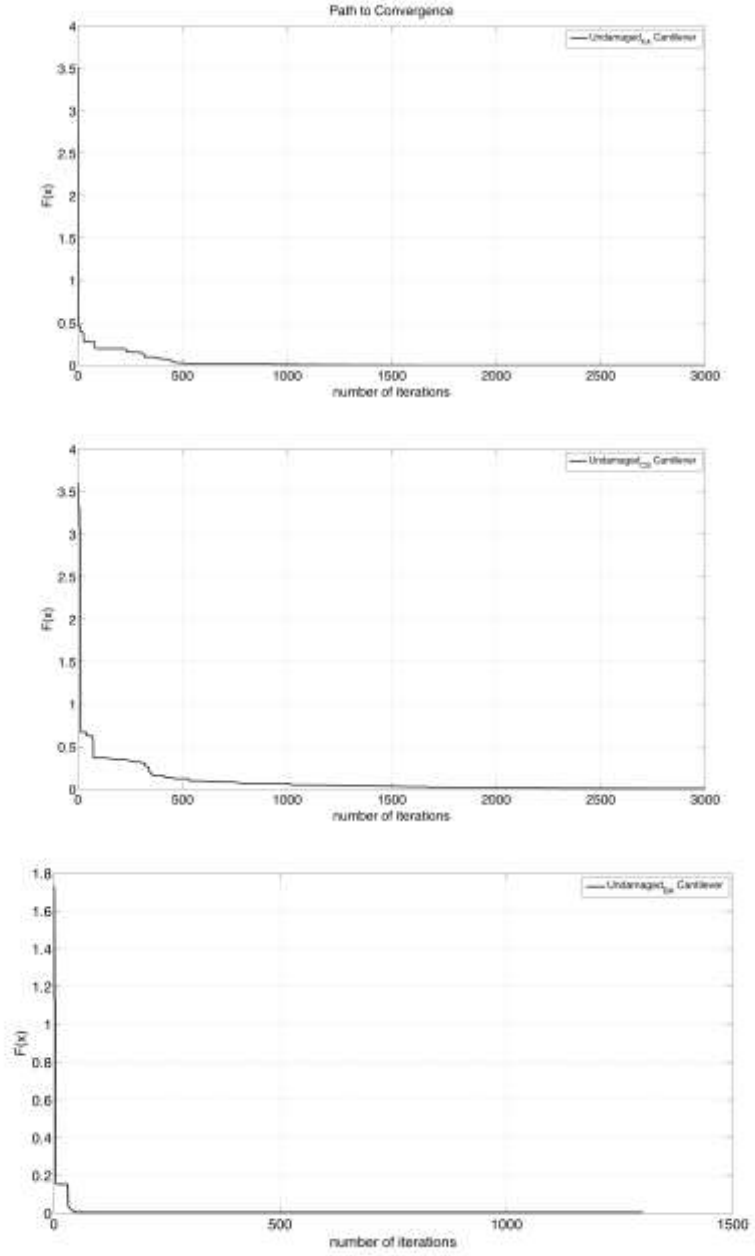


Figure 3.3 Path to convergence for *Undamaged-ABC Cantilever* (first), *Undamaged-FA Cantilever* (second), *Undamaged-CS Cantilever* (third), and *Undamaged-BA Cantilever* (fourth)

For sake of completeness, all the analyses carried out on a Mac OsX notebook, 64-bit, 2.8GHz Intel® Core i7 processor with 8GB ram.

3.2.2 Studies on damaged structures via ABC

As Table 3.2 reports, seven further analyses were carried out by applying the artificial bee colony algorithm, so localizing the damage in the structure under different scenarios. The first and second analyses on the damaged structure are labelled as *Damage A-ABC* and *Damage B-ABC* and seek the correct solution in element 1 and 9 respectively. These elements are close to the fixed edge of the beam. Figure 3.4 shows the paths to convergence for both cases, reporting the values of the objective function versus the number of iterations.

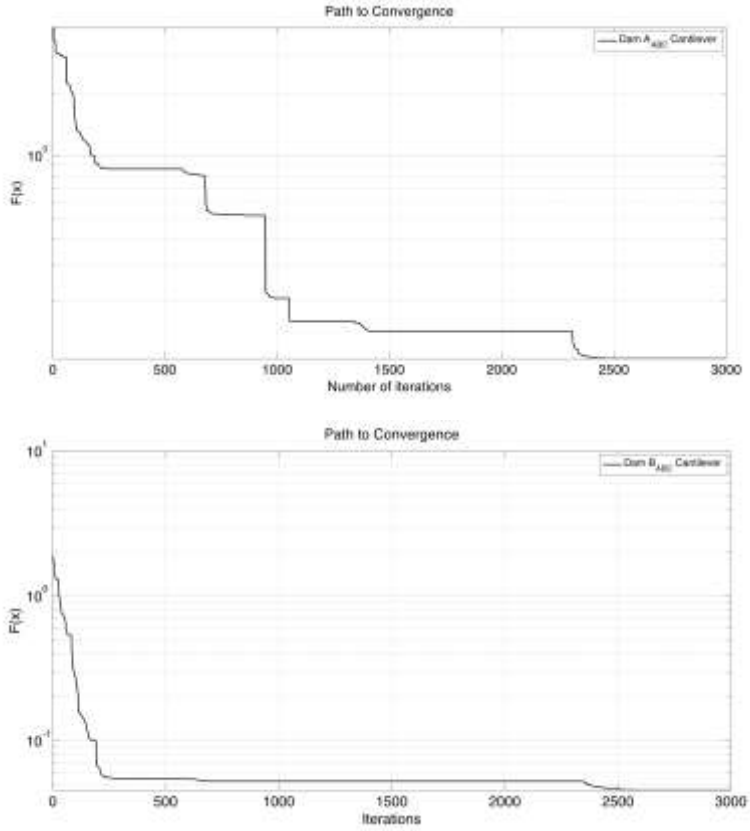


Figure 3.4 Path to convergence for cases *Damage A-ABC* (top) and *Damage B-ABC* (bottom)

Then, for the first case, the convergence is reached after 2519 iterations in 6429 seconds, while the second case converges after 2617 in 6033 seconds.
Multiple damages are introduced in case *Damage C-ABC* where a coefficient equal to 0.9 is assigned to both elements 1 and 9. For this case, after a running of 5060 seconds, the convergence is achieved in 1256 iterations.
When considering the case labelled as *Damage D-ABC* and *Damage E-ABC*, the assignment of the damage coincides with cases *A* and *B*, but the intensity is higher. Consequently, the value of null objective function is reached in 5103 seconds after 2684 iterations (case *Damage D-ABC*) and after 5039 seconds and 2959 iterations (case *Damage E-ABC*). For sake of completeness, Figure 3.5 and Figure 3.6 summarize the paths to convergence for cases from *Damage C-ABC* to *Damage F-ABC*.

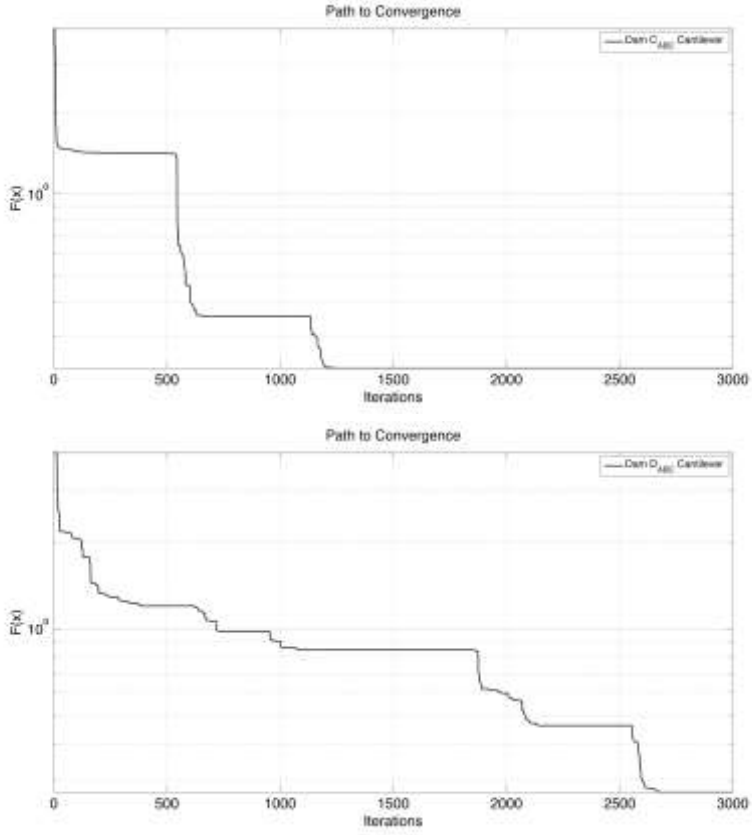


Figure 3.5 Path to convergence for cases *Damage C-ABC* (top) and *Damage D-ABC* (bottom)

In order to verify whether the behavior of central elements is the same as the edge ones, a small intensity of damage is introduced in the element 12, as shown in the recapitulatory Figure 3.2 (case *Damage F-ABC*). The convergence is reached after 2962 iterations in 5112 seconds.

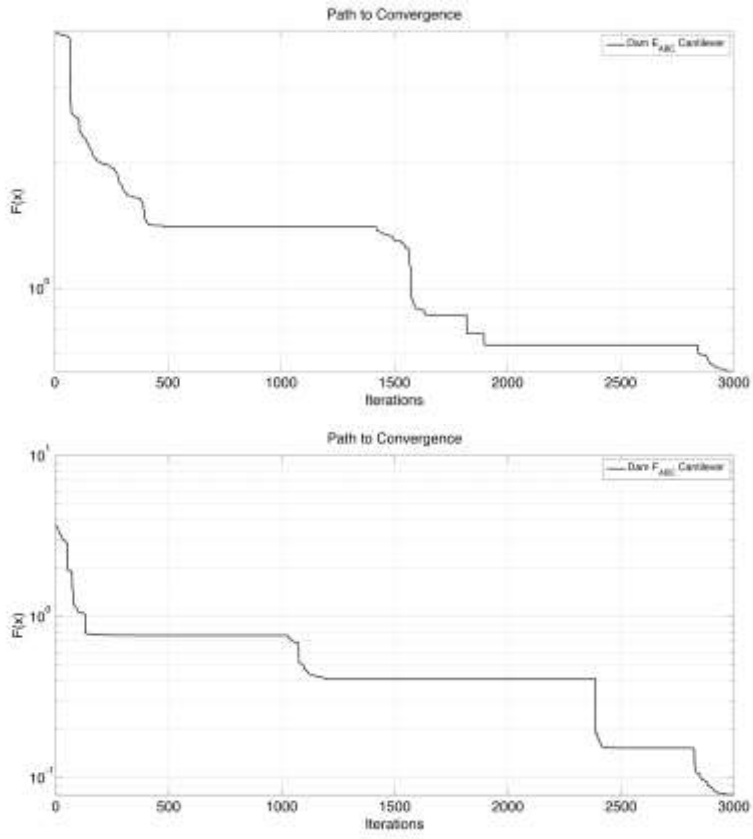


Figure 3.6 Path to convergence for cases *Damage E-ABC* (top) and *Damage F-ABC* (bottom)

Last analyzed analysis, namely *Damage-G-ABC*, is characterized by multiple, non-symmetric damage. In this case, the global optimality is reached in 5249 seconds after 2987 iterations. The path to convergence is shown in Figure 3.7.

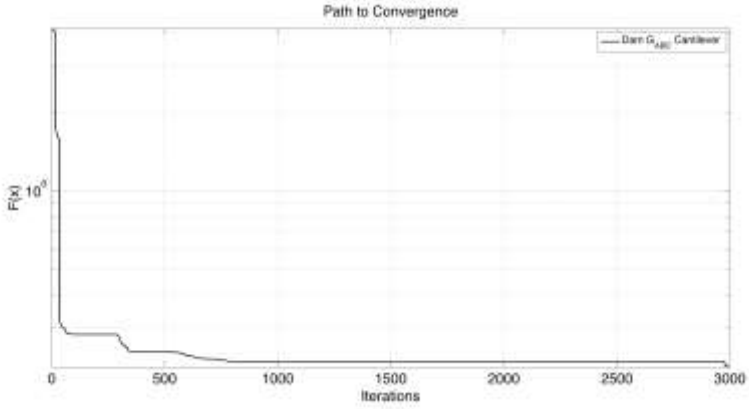


Figure 3.7 Path to convergence for case *Damage-G-ABC*

3.2.3 Studies on damaged structures via FA

In previous sub-section, the studies were focused on the identification of the stiffness matrices of the damage structure via ABC algorithm. The same analyses are carried out via firefly algorithm.

The first analysis, *Damage A-FA*, where damage is assigned in element 1, converges after 1543 iterations in 275 seconds.

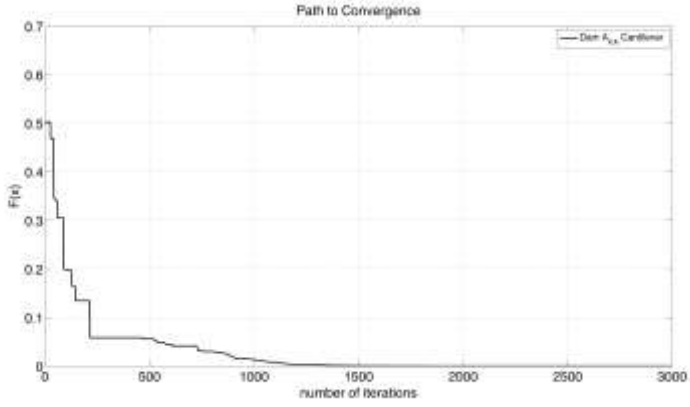


Figure 3.8 Path to convergence for case *Damage A-FA*

For sake of completeness, Figure 3.8 shows the path to convergence for the performed analysis and the typical shape of objective function is shown in Figure 3.9, setting the weight function 0 and 1, respectively in order to show the eigenvectors' contribution.

Indeed, when the weight is assigned to the objective function, the curve presenting a very steep slope in the neighborhood of the global minimum is obtained [6].

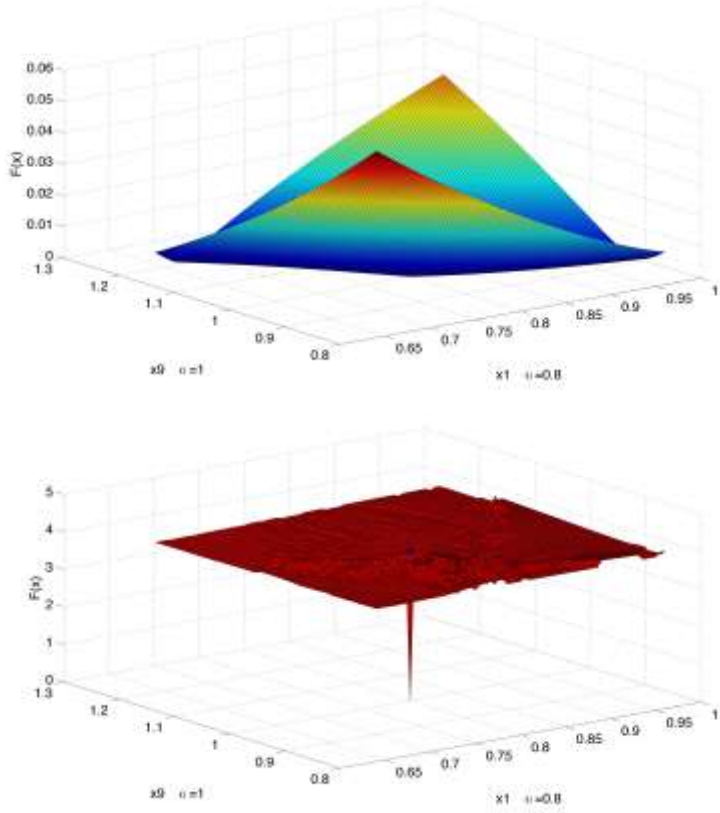


Figure 3.9 Shape of objective function for case *Damage A-FA* with $w = 0$ (top) and $w = 1$ (bottom)

In the bottom of Figure 3.9, the global optimum is located in a bottom of a 'well', and the challenge is entering in the 'well' without getting trapped on a local minimum. Indeed, it is obvious from the plots in Figure 3.9 that there exist a large amount of situations in which the requirements of the natural frequencies is satisfied, but only one point minimizes also the difference of the eigenvectors. Such point is the searched global minimum of the objective function, and it is reached once the current vector of the designed parameters corresponds to the current one. Moreover, the strength of the objective function formulation given in Section 3.1 in Equations (31) and (32) is represented by the existence and the uniqueness of the global minimum [6]. The second case (*Damage B-FA*) is quite similar to the first one, except that the damage is introduced in the upper element (the

ninth). Figure 3.10 shows that the convergence is achieved in 310 seconds after 1139 iterations.

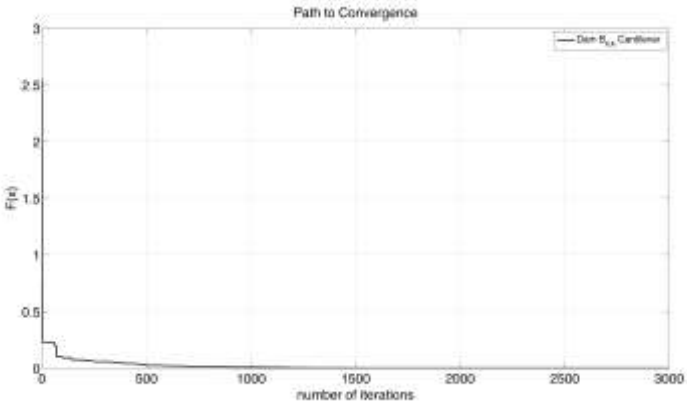


Figure 3.10 Path to convergence for case *Damage B-FA*

The third considered case is labelled *Damage C-FA*, in which the damage in introduced as symmetrical in both elements 1 and 9. For this case, the convergence is reached in 330 seconds after 1616 iterations and Figure 3.11 shows it.

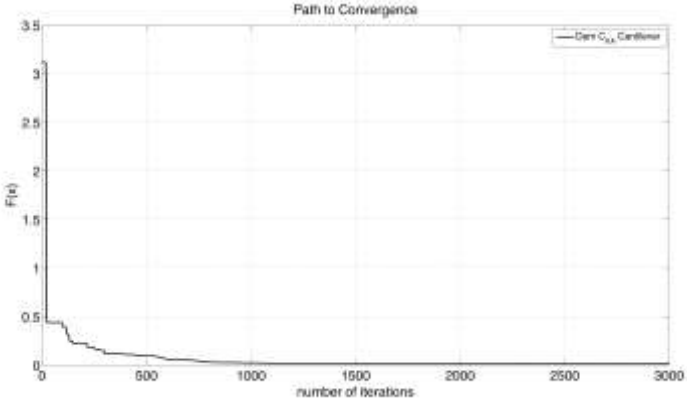


Figure 3.11 Path to convergence for case *Damage C-FA*

As in the previous Section, performing *Damage D-FA* and *Damage E-FA* the damage results in the same position as in cases *A* and *B*, but it has a different intensity, which changes from 0.8 to 0.4. The convergence is reached in 317 after 1231 iterations and in 331 seconds in 1421 iterations, respectively.

Hereafter, in Figure 3.12 the paths to convergence are summarized.

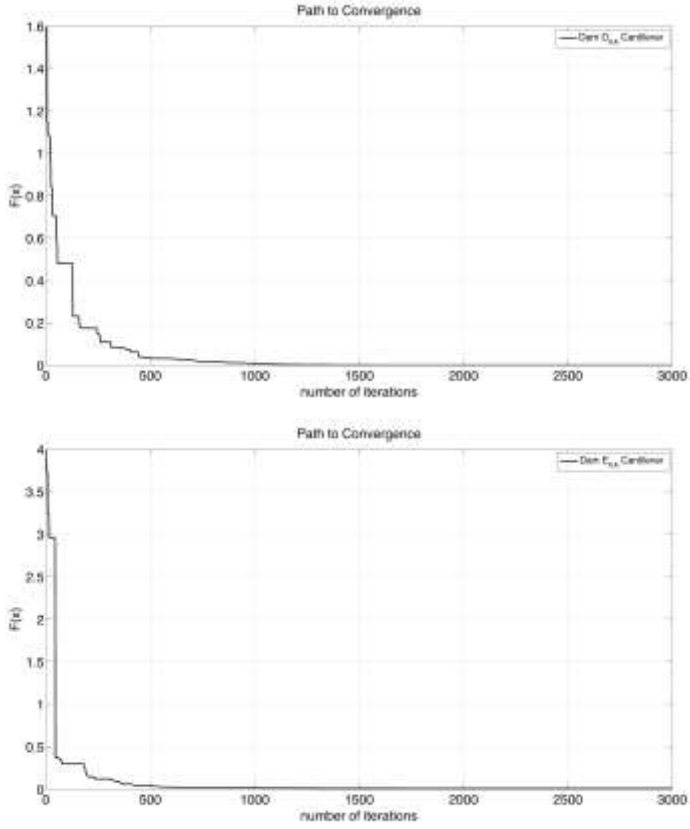


Figure 3.12 Path to convergence for cases *Damage D-FA* (top) and *Damage E-FA* (bottom)

Also here, for verifying that central elements behave as the edge ones, the damage is introduced in the twelfth element. The behavior is verified and here the convergence is achieved in 317 seconds after 1679 iterations. For this case the path to convergence is shown in Figure 3.13.

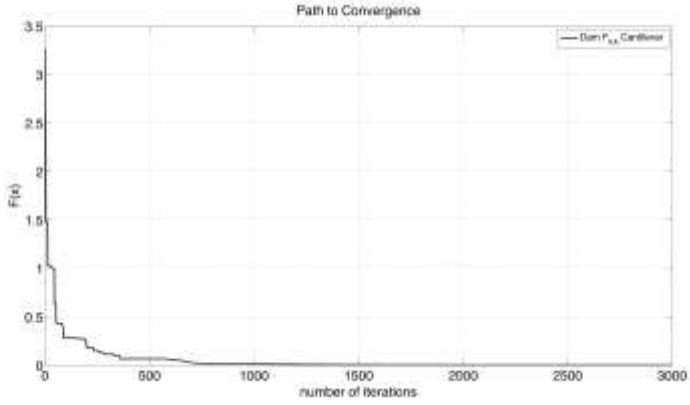


Figure 3.13 Path to convergence for case *Damage F-FA*

Last analyzed case introduced a non-symmetric and multiple damage in different elements (case *Damage G-FA*). Even there exist a highly nonlinearity, the convergence, as shown in Figure 3.14, is reached after 1430 for 323 seconds.

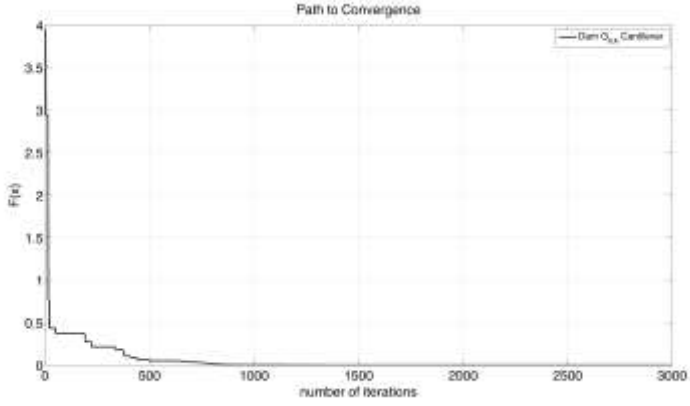


Figure 3.14 Path to convergence for case *Damage G-FA*

Due to the highly nonlinearity, Figure 3.15 reports the shape of the shape of the objective function, in which the uniqueness of the minimum solution is correctly achieved.

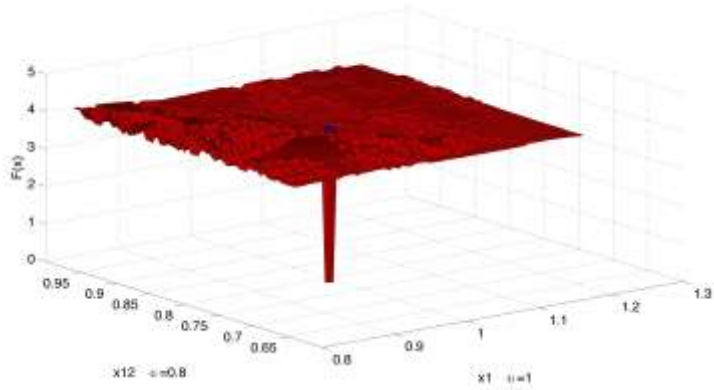
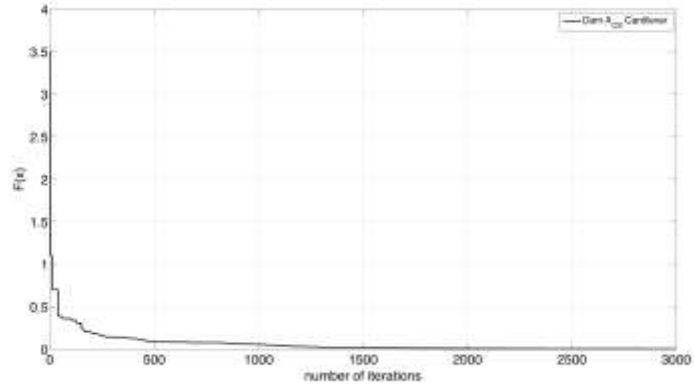


Figure 3.15 Shape of the objective function for case *Damage G-FA* with *w = 1*

3.2.4 Studies on damaged structures via CS

As in the last two sub-section, the analyses were carried out by exploiting the cuckoo search algorithm. The first and second analyses on the damaged structure (*Damage A-CS* and *Damage B-CS*) focus the research of the correct solution in elements 1 and 9, both close to the fixed edge of the beam. Figure 3.16 shows the paths to convergence for such cases, and gives the values of the objective function versus the number of iterations. Therefore, the convergence is reached after 2633 iterations in 594 seconds, while the second case converges after 2772 iterations in 498 seconds, respectively.



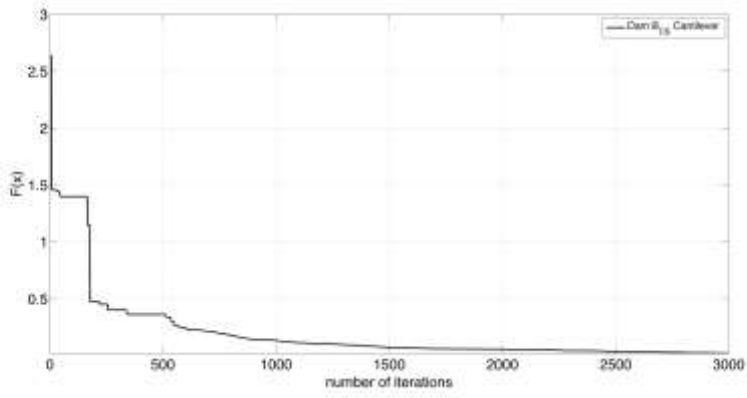
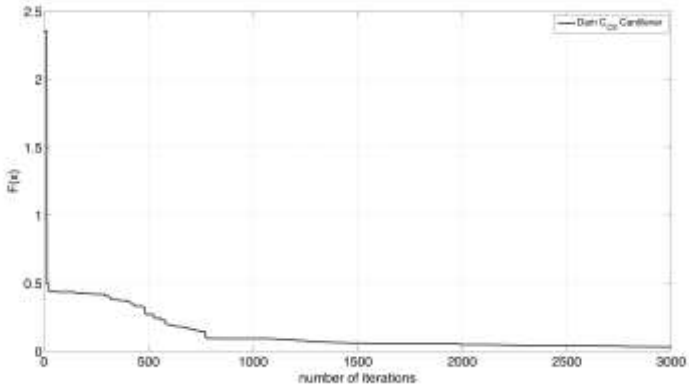


Figure 3.16 Path to convergence for cases *Damage A-CS* (top) and *Damage B-CS* (bottom)

Multiple damage is initialized in case *Damage C-CS* applying the same damage intensity to both elements considered in last two cases. For this case, the null solution arrives after 2554 iterations in 723 seconds.

Higher intensity of the damage ($\alpha = 0.4$) is assigned either in case *Damage D-CS* and *Damage E-CS*. Thus, the value of null objective function is reached in 745 seconds after 1877 iterations (case *Damage D-CS*) and after 615 seconds and 2841 iterations (case *Damage E-CS*). Both Figure 3.17 and Figure 3.18 illustrate the paths to convergence for cases from *Damage C-CS* to *Damage F-CS*.



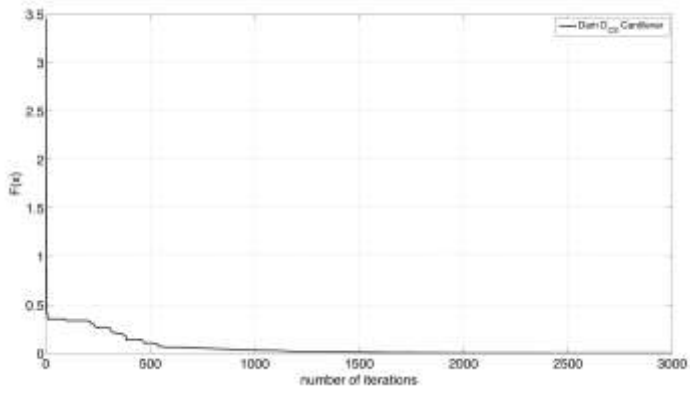
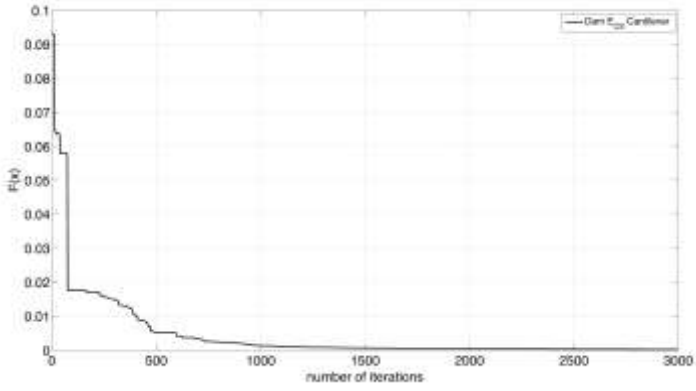


Figure 3.17 Path to convergence for cases *Damage C-CS* (top) and *Damage D-CS* (bottom)

A little intensity of damage is assigned to element 12 for verifying if the behavior is maintained the same (case *Damage F-CS*). Here, the convergence is reached after 2775 iterations in 617 seconds.



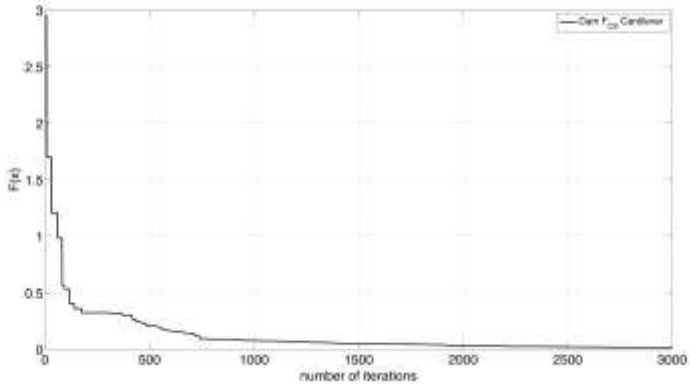


Figure 3.18 Path to convergence for cases *Damage E-CS* (top) and *Damage F-CS* (bottom)

Last performed analysis, *Damage-G-CS*, is characterized by multiple, non-symmetric damage. In this case, the global optimality is reached after 1556 iterations in 811 seconds, and the path to convergence is shown in Figure 3.19.

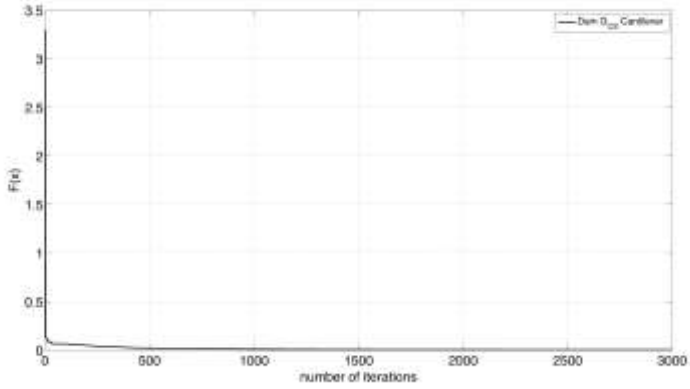


Figure 3.19 Path to convergence for case *Damage-G-CS*

3.2.5 Studies on damaged structures via BA

The same analyses were carried out via bat algorithm, as made in the other previous two sections. The first and second analyses (*Damage A-BA* and *Damage B-BA*) concentrate the search of the proper solution in elements 1 and 9, respectively. Figure 3.20 shows the paths to convergence for these cases, giving the values of the objective function versus

the number of iterations. Thus, the convergence is reached after 366 iterations in 757 seconds, while the second one reaches it after 358 iterations in 776 seconds, respectively.

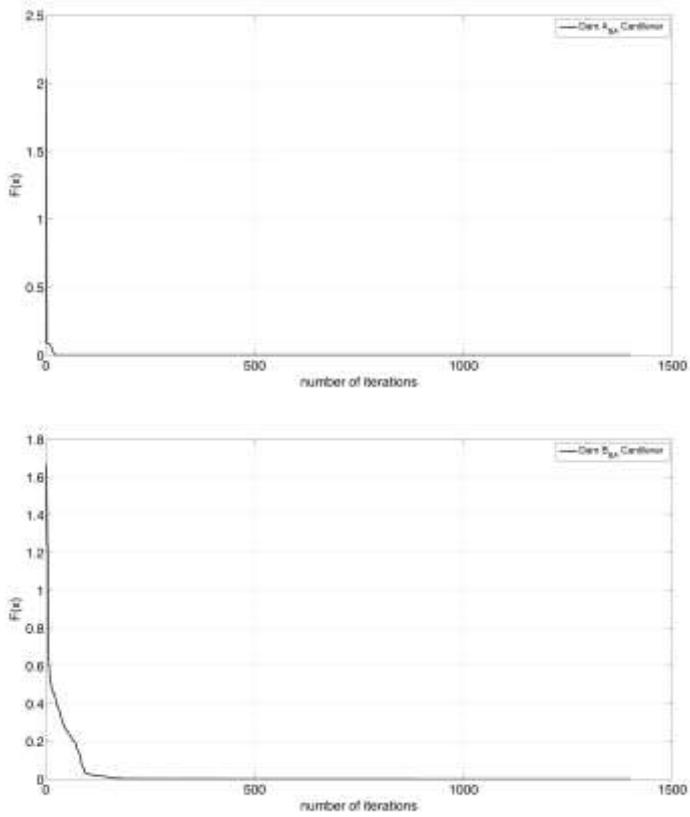


Figure 3.20 Path to convergence for cases *Damage A-BA* (top) and *Damage B-BA* (bottom)

Multiple damage is introduced in case *Damage C-BA* by the application of the same damage intensity to both elements considered in first and second analyses. Here, the null solution comes after 368 iterations in 595 seconds.

In both cases *Damage D-BA* and *Damage E-BA* a higher intensity of the damage is considered. Hence, the value of null fitness function is reached in 563 seconds after 307 iterations (case *Damage D-BA*) and after 481 seconds and 297 iterations (case *Damage E-BA*). Figure 3.21 and Figure 3.22 illustrate the paths to convergence for cases from *Damage C-BA* to *Damage F-BA*.

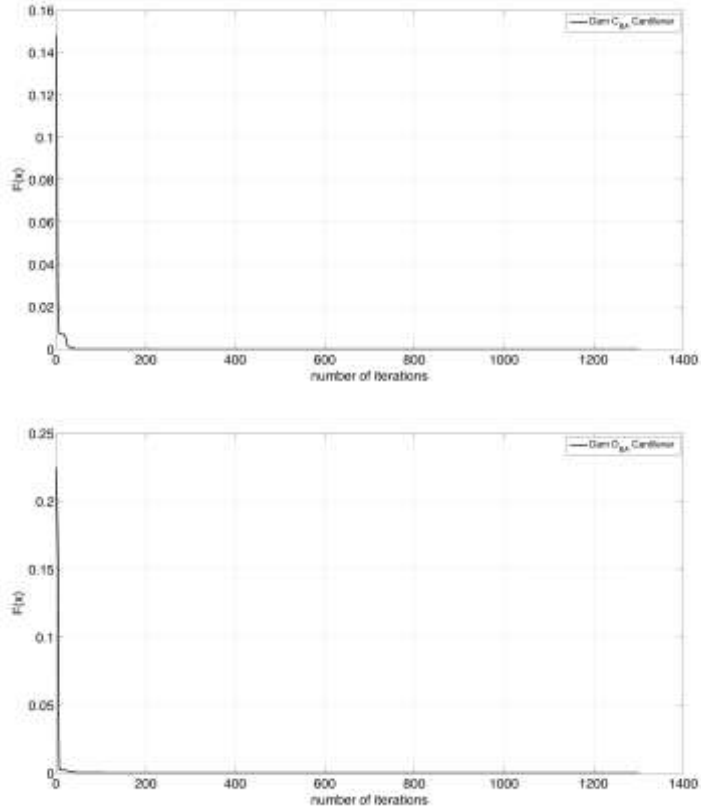


Figure 3.21 Path to convergence for cases *Damage C-BA* (top) and *Damage D-BA* (bottom)

In the sixth case the damage is assigned to element 12 (case *Damage F-BA*). Here, the convergence is reached after 319 iterations in 571 seconds.

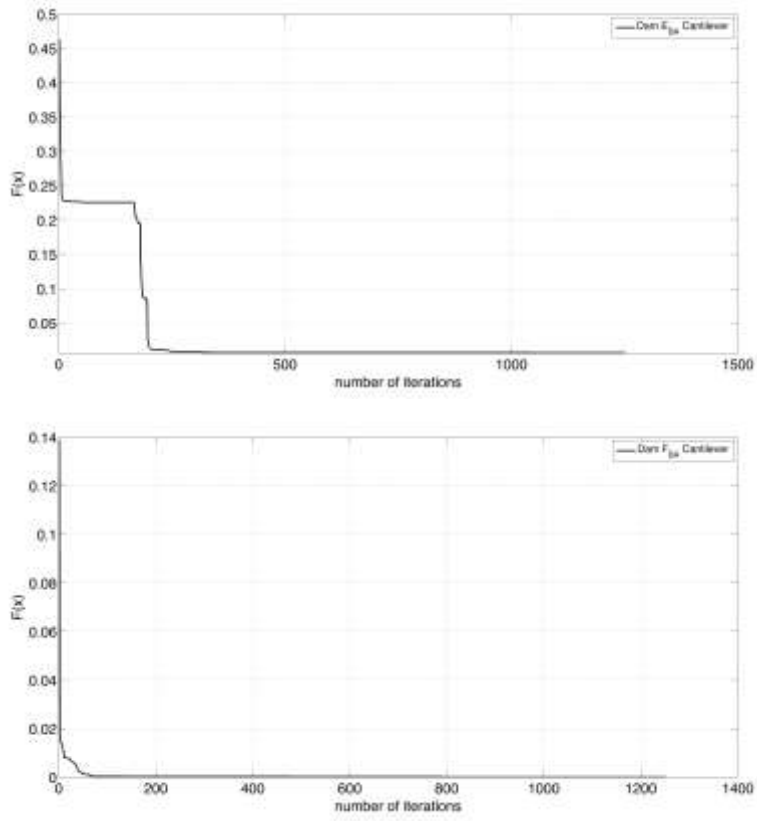


Figure 3.22 Path to convergence for cases *Damage E-BA* (top) and *Damage F-BA* (bottom)

Last performed analysis, namely *Damage-G-BA*, is outlined by multiple and non-symmetric damage. In this case, the global optimality is achieved after 301 iterations in 640 seconds and the path to convergence is shown in Figure 3.23.

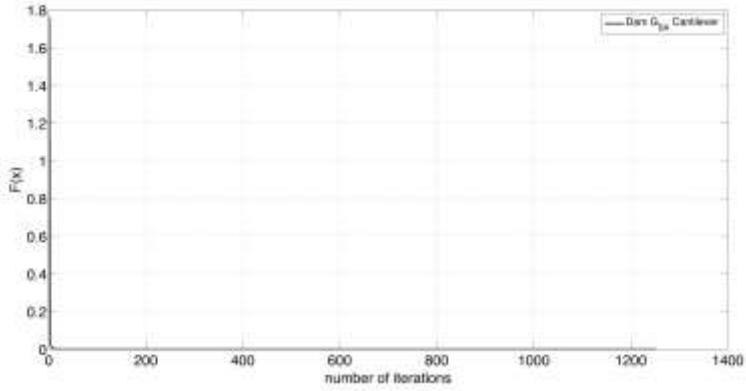


Figure 3.23 Path to convergence for case *Damage-G-BA*

3.2.6 Comparison between the methods

In this Sub-section, the performances of four metaheuristic tools, namely ABC, FA, CS and BA, that are employed in order to minimize the difference between the dynamic responses of a cantilever beam are compared. Once the damage scenario is introduced [11], [12] in a generic structure as a local stiffness deterioration [13], the adopted algorithms are able to converge and identify the correct damage scenario. The efficiency of the methods is evaluated in terms of number of iterations needed to converge and in the time duration of the analyses.

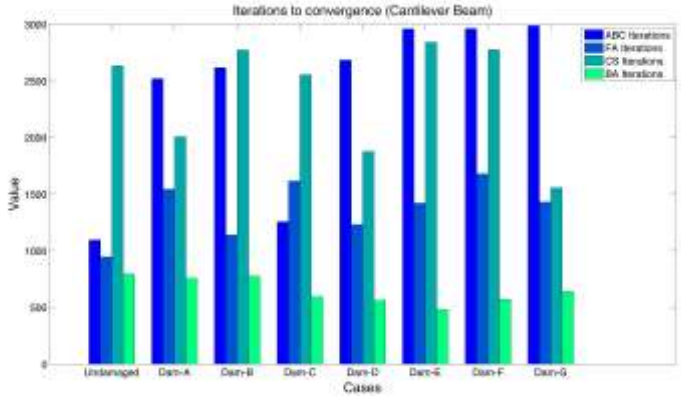


Figure 3.24 Number of iterations to converge (cantilever beam)

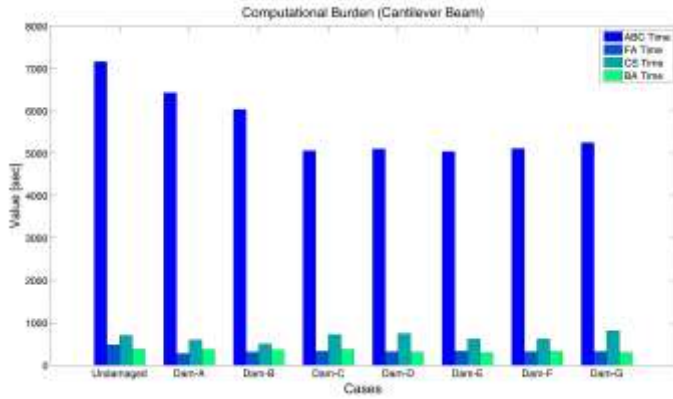


Figure 3.25 Time duration of the analyses (cantilever beam)

The calibration of the other parameters of each method is carried out in an initial step that was revealed crucial for equating them. As an attempt to summarize the results, the parameters are plotted in Figure 3.24 and Figure 3.25.
It is evident that the firefly algorithm is more efficient in terms of duration time of the analysis, while the bat algorithm is quicker in finding the best optima in a limited number of iterations. It is worth noticing that the duration time for the FA is almost twenty times smaller than the time employed by the ABC. The performance of the CS are quite satisfactory in terms of computational burden and comparable with the FA ones.

3.3 The frame structure

The same optimization problem performed in Section 3.2 is herein carried out, but applied on civil structures [9] as two-dimensional truss with 23 elements and 13 nodes as stated in Figure 3.26.

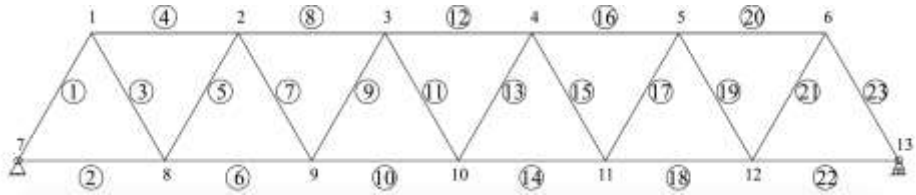


Figure 3.26 Discretization of the frame structure

Two structural scenarios are imagined in which each element is either fixed or hinged to each other. These two configurations will be labelled as *Undamaged Frame-Fix* and *Undamaged Frame-Hin*, respectively. The truss is simply supported, thus within a finite elements framework, the number of degrees of freedom changes from 36 to 23, depending on the structural configuration considered. The material is isotropic and on each element Young modulus, Poisson ratio, mass density, length, and cross-section area are set. Table 3.8 summarizes such features.

Table 3.8 Features of the frame structure (fixed/hinged configurations)

Property	Value
Length (l_{el})	1m
Cross-section area (A_{el})	0.004m^2
Number of nodes per element (n_e)	2
Number of elements (m)	27
Number of nodes (n)	13
Number of degrees of freedom	36/23
Young modulus (E)	$2.0\times10^{11}\ \text{N}/\text{m}^2$
Poisson ratio (ν)	0.2
Mass density (ρ)	$7.8\times10^{7}\ \text{kg}/\text{m}^3$

The same approach described in Section 3.2 has been adopted for introducing the damage in the e-th element. Table 3.9 and Figure 3.27 show the damage scenarios for each structural configuration herein considered.

Table 3.9 Different structural configurations and damage scenarios (frame structure)

Structural configuration	Damaged element(s)	α_e
Undamaged	-	1
Damage 1 (D1)	2	0.8
Damage 2 (D2)	3	0.8
Damage 3 (D3)	2 - 3	0.8
Damage 4 (D4)	15	0.5
Damage 5 (D5)	2 – 4 – 20 – 22	0.8 – 0.8 – 0.8 – 0.8
Damage 6 (D5)	2 – 4 – 20 - 22	0.8 – 0.7 – 0.9 – 0.8

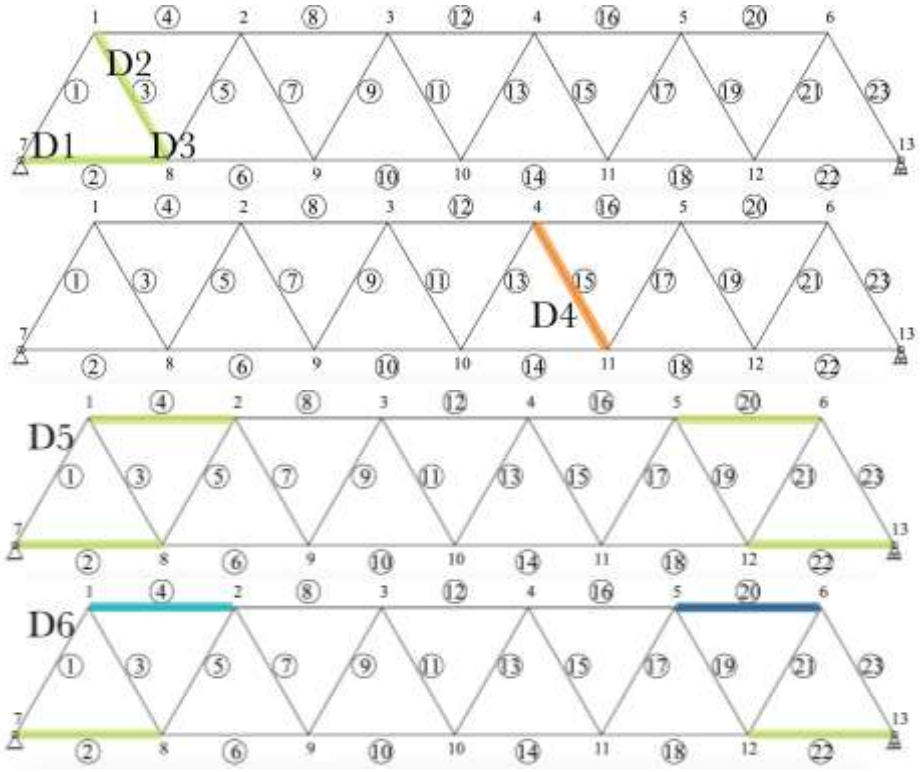


Figure 3.27 Different damage scenarios for each structural configuration (frame structure)

By the solution of the eigenvalues-eigenvectors problem of Equation (30), the first nine eigenvalues of each structural scenario are reported in Table 3.10 and Table 3.11, respectively.

Table 3.10 First exact nine modal frequencies for different values of element stiffness coefficient (truss with fixed elements)

Structural configuration	Exact values of first 9 circular frequencies (rad/s)								
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
Undamaged FF	602.3	880.1	957.4	980.2	1058.1	1101.7	1242.2	1254.1	1386.4
Damage 1 (D1-FF)	588.3	860.3	951.5	972.0	1058.1	1089.2	1237.8	1243.7	1375.9
Damage 2 (D2-FF)	601.9	879.2	923.0	972.2	1051.1	1100.5	1238.3	1253.3	1385.6
Damage 3 (D3-FF)	587.5	859.0	912.6	969.5	1050.8	1088.8	1235.6	1242.1	1375.1
Damage 4 (D4-FF)	600.9	877.7	925.5	967.1	1035.5	1105.3	1205.5	1252.2	1381.2
Damage 5 (D5-FF)	577.8	840.9	948.0	958.6	1057.8	1062.3	1189.1	1237.9	1324.9
Damage 6 (D6-FF)	575.7	839.6	948.0	958.3	1057.9	1060.4	1186.3	1238.4	1325.7

Table 3.11 First exact nine modal frequencies for different values of element stiffness coefficient (truss with hinged elements)

Structural Configuration	Exact values of first 9 circular frequencies (rad/s)								
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
Undamaged FH	140.1	172.4	192.0	212.0	223.2	246.1	275.1	283.7	317.6
Damage 1 (D1-FH)	139.9	171.8	191.8	211.2	222.2	245.7	273.6	282.4	317.5
Damage 2 (D2-FH)	140.0	172.0	191.9	211.7	222.8	245.6	274.5	283.2	316.7
Damage 3 (D3-FH)	139.8	171.4	191.6	210.9	221.5	245.1	272.8	281.9	316.5
Damage 4 (D4-FH)	137.0	170.4	186.7	209.3	221.3	242.1	273.7	278.2	310.1
Damage 5 (D5-FH)	137.5	169.6	187.7	206.5	217.8	242.5	266.8	275.5	315.2
Damage 6 (D6-FH)	137.6	169.1	188.1	206.5	217.9	243.9	266.9	277.5	315.5

The quantities $\bar{\boldsymbol{\omega}}_{exact}$ and $\bar{\boldsymbol{\Phi}}_{exact}$ have size $n_{dof} \times 1$ and $n_{dof} \times n_{dof}$ respectively. In each performed analysis, both quantities are utilized as input parameters of the chosen algorithm. Then, the weight function, namely *w*, has been set 0 and 1 in order to highlight or not the eigenvectors' contribution.

Theoretically, the existence domain of each design variable is included in the defined interval, and for better framing the adequate search domain, a preliminary study on the undamaged structure is required. Such an operation is useful both for facilitating the convergence of the method and for calibrating the control parameters of the solving algorithms. Consequently, the damage detection and localization is pursued as stated in Table 3.9.

3.3.1 Preliminary analyses on undamaged structures

First, the control parameters of the applied optimization tools are reported in Tables from Table 3.12 to Table 3.19.

Table 3.12 Control parameters of ABC (frame structure with fixed elements)

Control parameters of ABC	Values for each structural configuration					
	$D1_{ABC}$-FF	**$D2_{ABC}$-FF**	**$D3_{ABC}$-FF**	**$D4_{ABC}$-FF**	**$D5_{ABC}$-FF**	**$D6_{ABC}$-FF**
CS, size of the initial population of honeybees	46	46	46	46	46	46
C_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
λ, limit number for abandoning the food search	100	100	100	100	100	100

Table 3.13 Control parameters of ABC (frame structure with hinged elements)

Control parameters of ABC	Values for each structural configuration					
	$D1_{ABC}$-FH	**$D2_{ABC}$-FH**	**$D3_{ABC}$-FH**	**$D4_{ABC}$-FH**	**$D5_{ABC}$-FH**	**$D6_{ABC}$-FH**
CS, size of the initial population of honeybees	46	46	46	46	46	46
C_{max}, maximum Number of iterations	1500	1500	1500	1500	1500	1500
λ, limit number for abandoning the food search	100	100	100	100	100	100

Table 3.14 Control parameters of FA (frame structure with fixed elements)

Control parameters of FA	Values for each structural configuration					
	$D1_{FA}$-FF	**$D2_{FA}$-FF**	**$D3_{FA}$-FF**	**$D4_{FA}$-FF**	**$D5_{FA}$-FF**	**$D6_{FA}$-FF**
NP, size of the initial population of fireflies	100	100	100	100	100	80
I_{max}, maximum number of iterations	2000	2000	2000	2000	2000	2000
ζ, randomization number	0.5	0.5	0.5	0.5	0.5	0.5
β_{min}, minimum attractiveness	0.2	0.2	0.2	0.2	0.2	0.2
γ, absorption coefficient	1.0	1.0	1.0	1.0	1.0	1.0

Table 3.15 Control parameters of FA (frame structure with hinged elements)

Control parameters of FA	Values for each structural configuration					
	$D1_{FA}$-FH	**$D2_{FA}$-FH**	**$D3_{FA}$-FH**	**$D4_{FA}$-FH**	**$D5_{FA}$-FH**	**$D6_{FA}$-FH**
NP, size of the initial population of fireflies	80	80	80	80	80	80
I_{max}, maximum number of iterations	1500	1500	1500	1500	1500	1500
ζ, randomization number	0.5	0.5	0.5	0.5	0.5	0.5
β_{min}, minimum attractiveness	0.2	0.2	0.2	0.2	0.2	0.2
γ, absorption coefficient	1.0	1.0	1.0	1.0	1.0	1.0

Table 3.16 Control parameters of CS (frame structure with hinged elements)

Control parameters of CS	Values for each structural configuration					
	$D1_{FA}$-FH	**$D2_{FA}$-FH**	**$D3_{FA}$-FH**	**$D4_{FA}$-FH**	**$D5_{FA}$-FH**	**$D6_{FA}$-FH**
n, number of nests	40	40	40	40	40	40
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
p_a, discovery rate	0.25	0.25	0.25	0.25	0.25	0.25

Table 3.17 Control parameters of CS (frame structure with hinged elements)

Control parameters of CS	Values for each structural configuration					
	$D1_{FA}$-FH	**$D2_{FA}$-FH**	**$D3_{FA}$-FH**	**$D4_{FA}$-FH**	**$D5_{FA}$-FH**	**$D6_{FA}$-FH**
n, number of nests	30	30	30	30	30	30
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
p_a, discovery rate	0.25	0.25	0.25	0.25	0.25	0.25

Table 3.18 Control parameters of BA (frame structure with hinged elements)

Control parameters of BA	Values for each structural configuration					
	$D1_{FA}$-FH	**$D2_{FA}$-FH**	**$D3_{FA}$-FH**	**$D4_{FA}$-FH**	**$D5_{FA}$-FH**	**$D6_{FA}$-FH**
n, size of the initial population of bats	100	1000	100	100	100	100
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
loudness	1	1	1	1	1	1
pulse rate	0.8	0.8	0.8	0.8	0.8	0.8

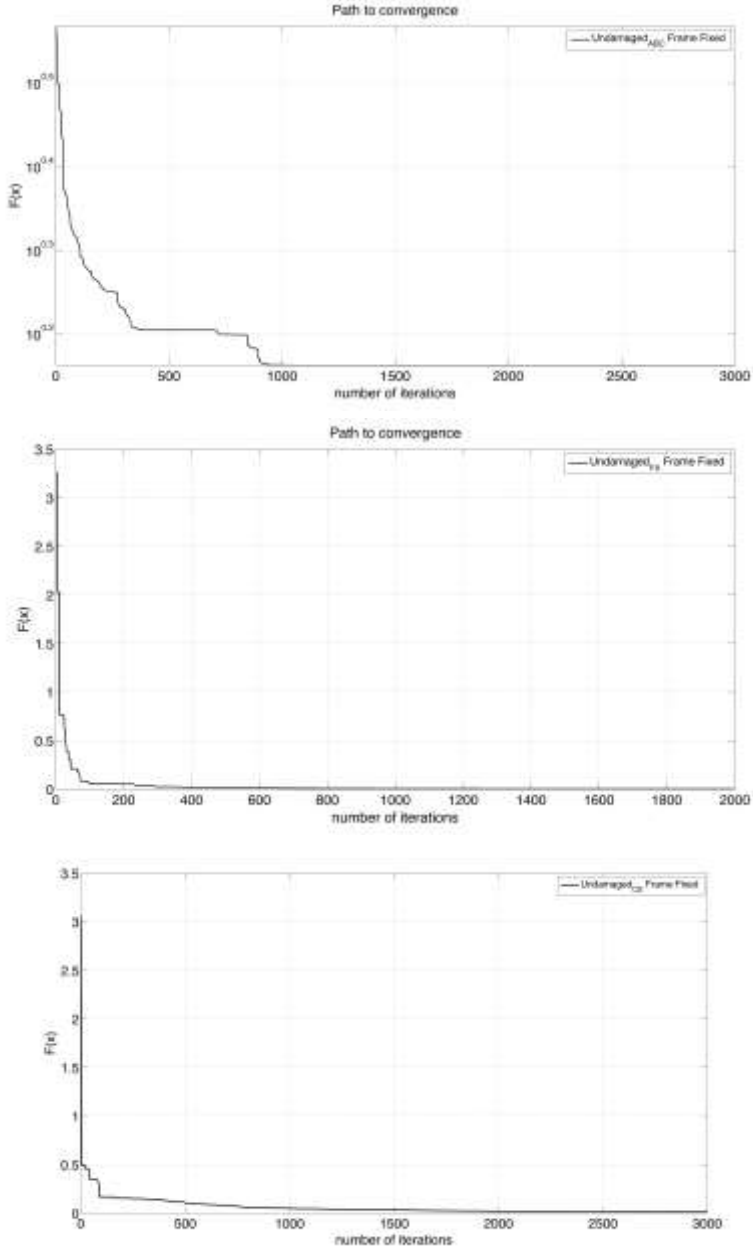
Table 3.19 Control parameters of BA (frame structure with hinged elements)

Control parameters of BA	Values for each structural configuration					
	$D1_{FA}$-FH	**$D2_{FA}$-FH**	**$D3_{FA}$-FH**	**$D4_{FA}$-FH**	**$D5_{FA}$-FH**	**$D6_{FA}$-FH**
n, size of the initial population of bats	80	80	80	80	80	80
I_{max}, maximum number of iterations	2000	2000	2000	2000	2000	2000
loudness	1	1	1	1	1	1
pulse rate	0.8	0.8	0.8	0.8	0.8	0.8

Also there, the search domain is centered around the value x_0, assigning as an interval in which the value is increased and decreased by the 20%.

After these assumptions are fixed, the convergence, for the structural scenario labelled as *Undamaged Frame-Fix*, the solution is achieved in 8849 seconds after 931 iterations by ABC, in 427 seconds after 783 iterations by FA, in 493 seconds after 2542 iterations by CS, while in 698 seconds in 1704 iterations by BA.

Similar results are obtained for the other undamaged scenario, namely *Undamaged Frame-Hin*: the solution is reached in 193 seconds after 940 iterations by FA, in 262 seconds after 2276 seconds by CS, in 3201 seconds after 506 iterations by ABC, and in 242 seconds after 1253 iterations by BA.


Path to convergence
F(x)
number of iterations
Path to convergence
F(x)
number of iterations
F(x)
number of iterations

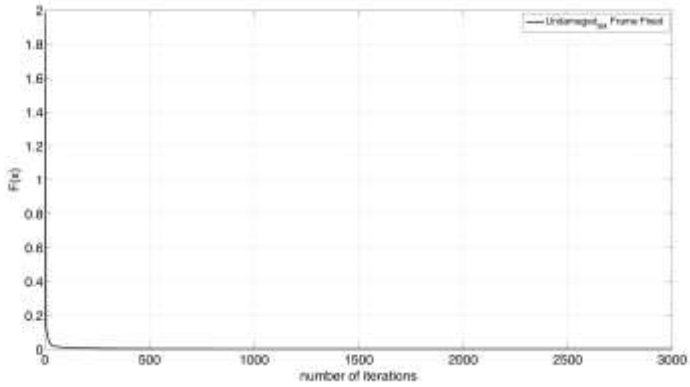
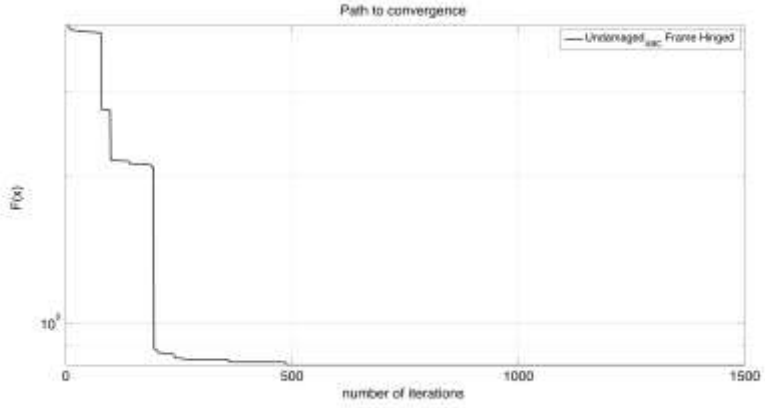


Figure 3.28 Path to convergence for cases *Undamaged ABC Frame-Fix* (first), *Undamaged FA Frame-Fix* (second), *Undamaged CS Frame-Fix* (third), *Undamaged BA Frame-Fix* (fourth)

Both Figure 3.28 and Figure 3.29 present the path to convergence of undamaged cases.



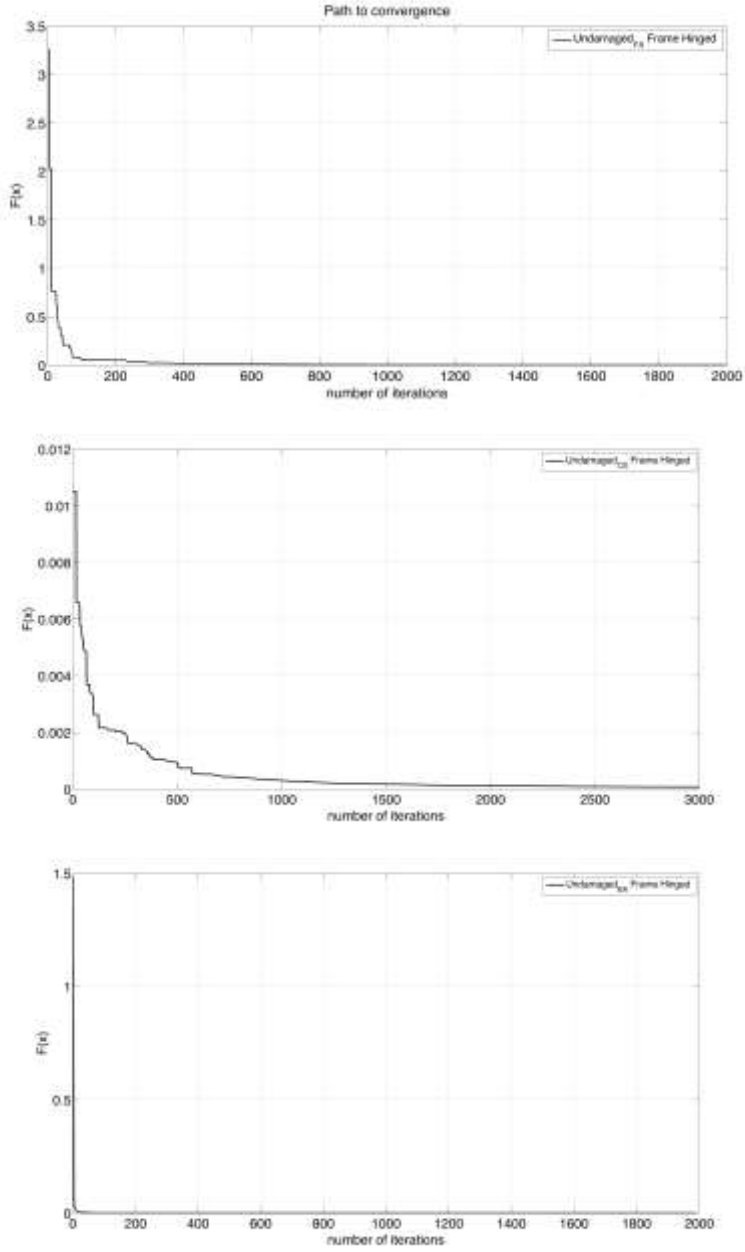
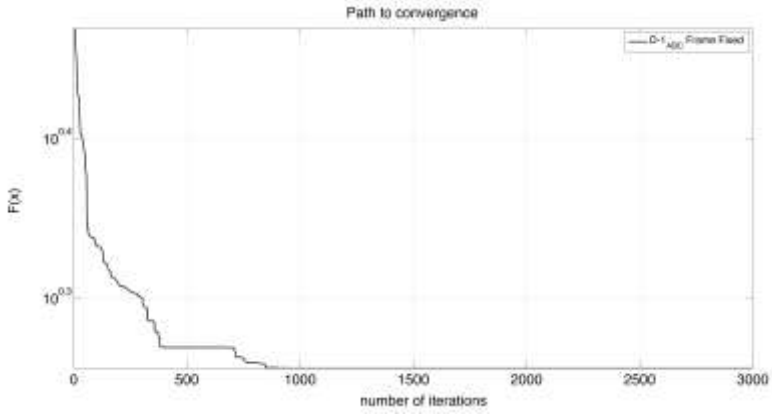


Figure 3.29 Path to convergence for cases *Undamaged ABC Frame-Hin* (first), *Undamaged FA Frame-Hin* (second), *Undamaged CS Frame-Hin* (third), and *Undamaged BA Frame-Hin* (fourth)

Hence, all the algorithms converge but the FA is quicker than the ABC, the CS (particularly in terms of duration of the analysis), and the BA.
For sake of completeness, also this set of analyses is carried out on a Mac OsX notebook, 64-bit, 2.8GHz Intel® Core i7 processor with 8GB ram.

3.3.2 Studies on damaged structures via ABC

For detecting and localizing the damage in the truss structure under different scenarios (see Table 3.9), six further analyses were performed by applying the artificial bee colony algorithm. First, the structural configuration with fixed element is considered, and then the same analyses are repeated introducing the internal hinges in the structure.
The first analyses focuses the research in element 2, i.e. the first horizontal element close to the support. Performing the analysis on the fixed configuration the convergence is reached in 8470 seconds after 899 iterations (*D1-ABC-FF*), while with hinges (*D1-ABC-FH*) the algorithm converges in 3223 seconds with 416 iterations. Figure 3.30 draws the situation just described.



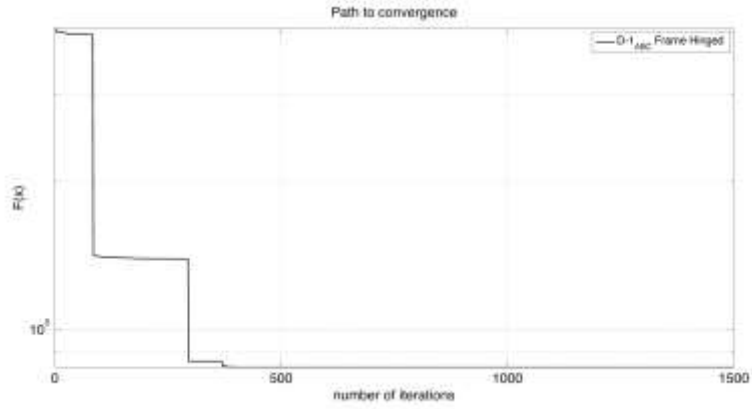
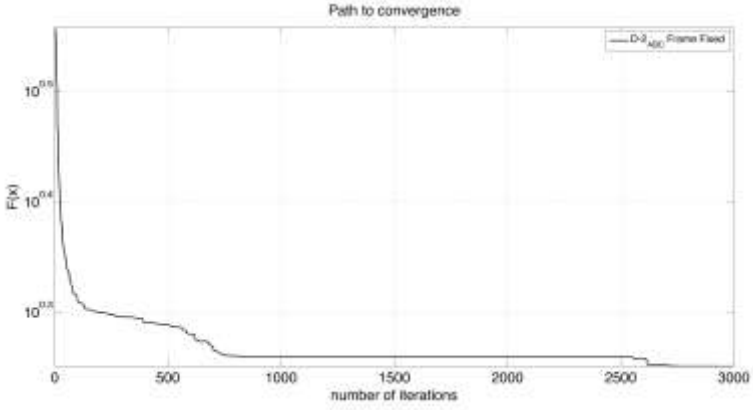


Figure 3.30 Path to convergence for cases *D1-ABC-FF* (top) and *D1-ABC-FH* (bottom)

The second cases, namely *D2-ABC-FF* and *D2-ABC-FH*, are similar to the first one in terms of damage intensity, but this time the damaged element is the third one, a transversal one. The convergence, as shown in Figure 3.31, is achieved after 2653 iterations in 7930 seconds for the structure with fixed elements and in 2705 seconds after 382 iterations.



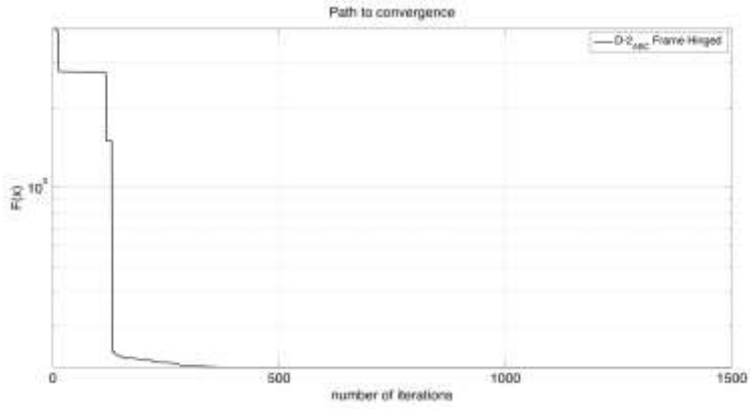
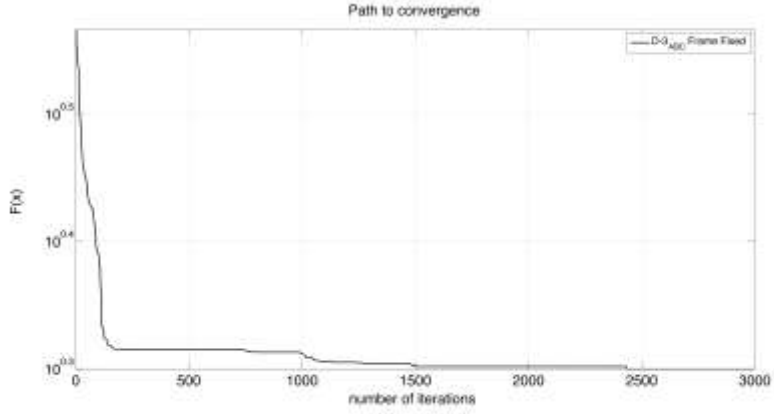


Figure 3.31 Path to convergence for cases *D2-ABC-FF* (top) and *D2-ABC-FH* (bottom)

The third case (*D3-ABC-FF* and *D3-ABC-FH*) introduces multiple damage in elements 2 and 3 and the solutions are achieved in 2433 iterations with a computational time equal to 9960 seconds, and in 172 iterations in 3773 seconds, respectively. Again, for sake of comparison, the results obtained via ABC from the truss with fixed and hinged elements are shown in Figure 3.32.



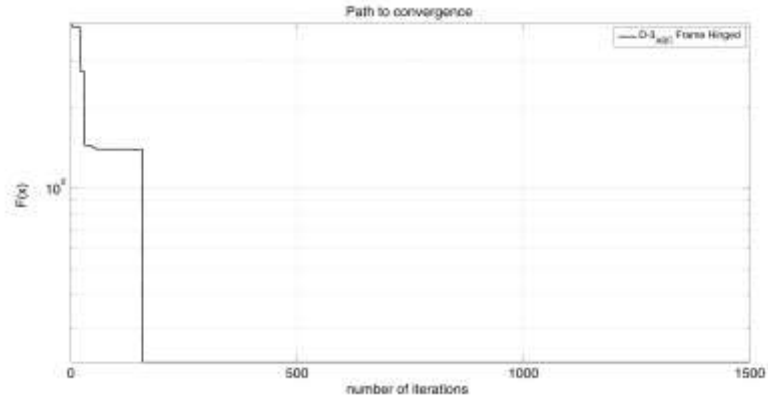
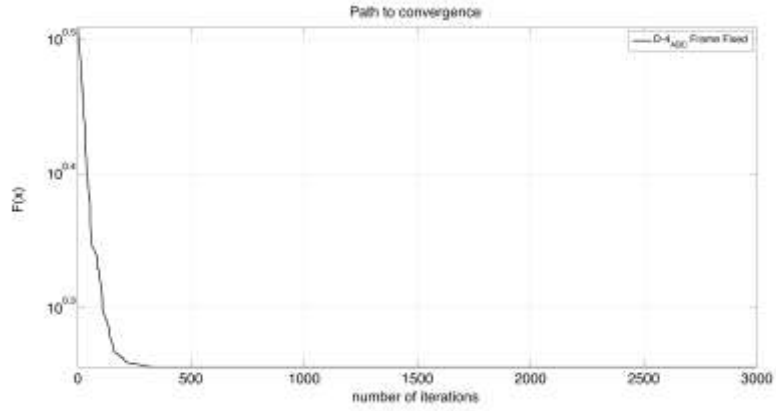


Figure 3.32 Path to convergence for cases *D3-ABC-FF* (top) and *D3-ABC-FH* (bottom)

The fourth case, labelled as *D4-ABC-FF* and *D4-ABC-FH*, is intended to verify the behavior in a central element. Thus, the damage is introduced in element 15 and the convergence is reached in 8891 seconds after 331 seconds. For the same structure with hinged elements, the null solution is found after 226 iterations with a computational time of 3863 seconds. Also for these cases, the comparison between the results is drawn in Figure 3.33.



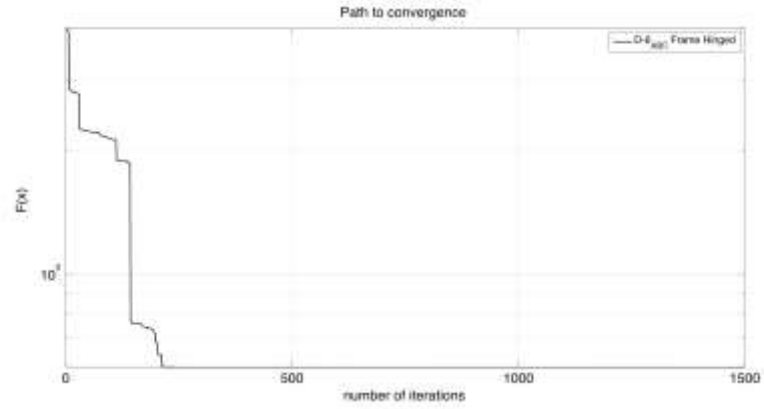
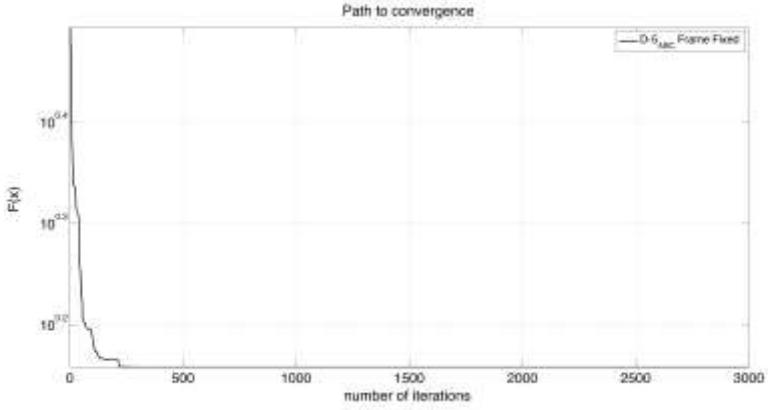


Figure 3.33 Path to convergence for cases *D4-ABC-FF* (top) and *D4-ABC-FH* (bottom)

Last two cases, namely *D5-ABC-FF*, *D5-ABC-FH*, *D6-ABC-FF*, and *D6-ABC-FH*, are realized to investigate the multiple symmetric and non-symmetric damage, respectively, by assigning the damage to elements 2 – 4 – 20 – 22, and varying its intensity.
For the fitht cases, the analyses converge in 310 iterations in 8677 seconds and after 641 iterations in 3106 seconds, respectively. Considering the last case, the analysis performed with fixed elements reaches a null solution in 8964 seconds after 641 seconds, while with the hinged ones the null solution is achieved in 289 iterations in 2712 seconds.
The paths to convergence of these cases are shown in Figure 3.34 and Figure 3.35.



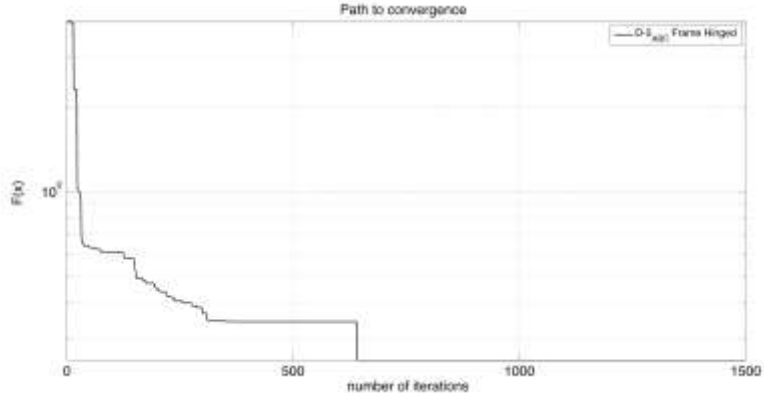


Figure 3.34 Path to convergence for cases *D5-ABC-FF* (top) and *D5-ABC-FH* (bottom)

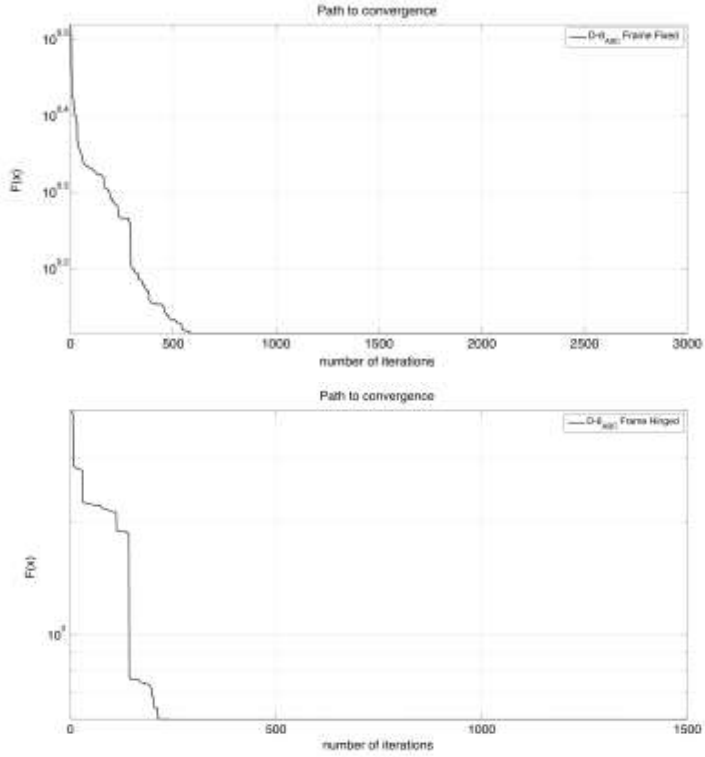


Figure 3.35 Path to convergence for cases *D6-ABC-FF* (top) and *D6-ABC-FH* (bottom)

3.3.3 Studies on damaged structures via FA

Same analyses, performed in the previous Section, were replayed with the FA. Once again, the structure with fixed elements is initially considered, and then the internal hinges are introduced.

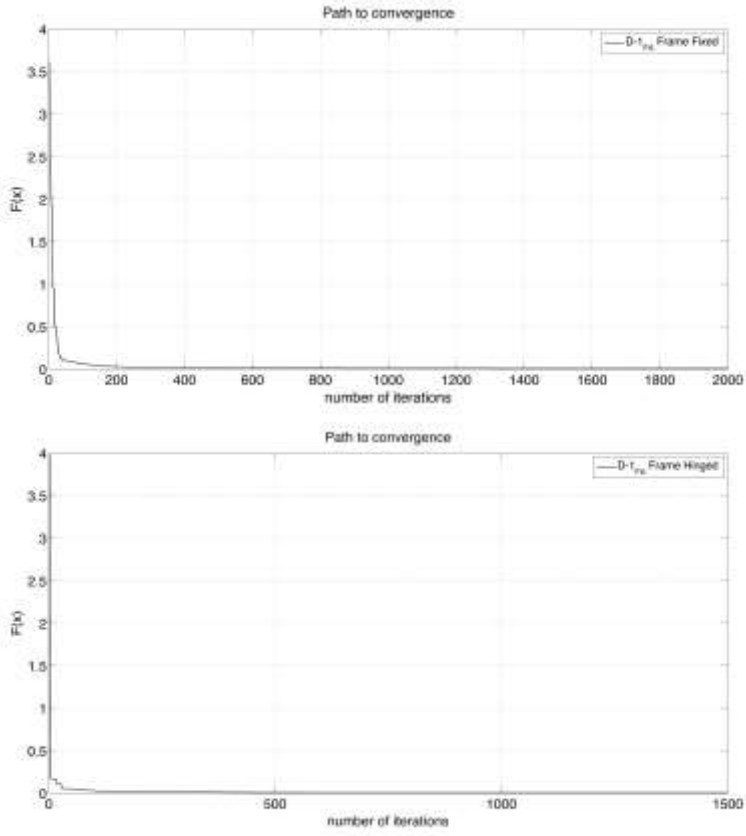


Figure 3.36 Path to convergence for cases *D1-FA-FF* (top) and *D1-FA-FH* (bottom)

The first analysis (*D1-FA-FF*) which assigns the damage in the second element, reaches the convergence after 1030 iterations in 420 seconds. The same structure, but hinged (*D1-FA-FH*), converges after 634 iterations in 192 seconds. Figure 3.36 shows the paths to convergences for both cases. Furthermore, for sake of exemplification, the shape of the objective function is presented in Figure 3.37.

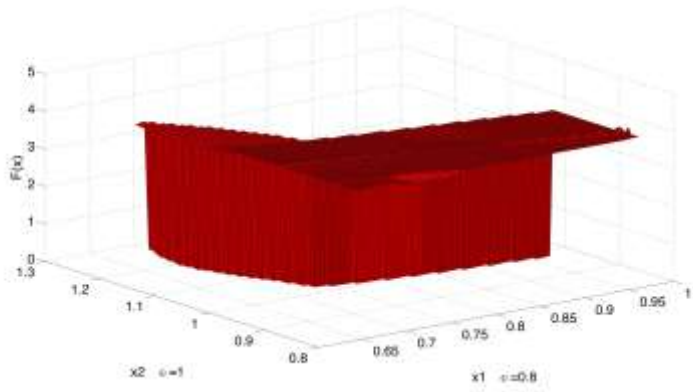
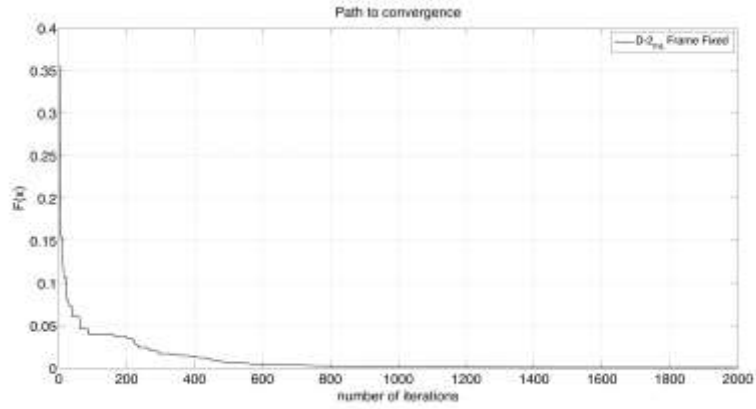


Figure 3.37 Shape of objective function for case *D1-FA-FF* with *w* = 1

The second case (*D2-FA-FF* and *D2-FA-FH*) aims to detect the damage in the third element and it converges in 617 seconds after 1126 iterations and in 166 seconds after 704 iterations, respectively. Figure 3.38 presents the path to convergence in both cases.



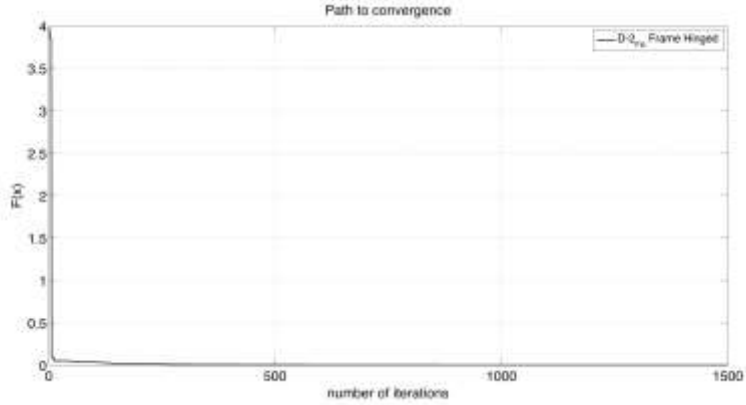
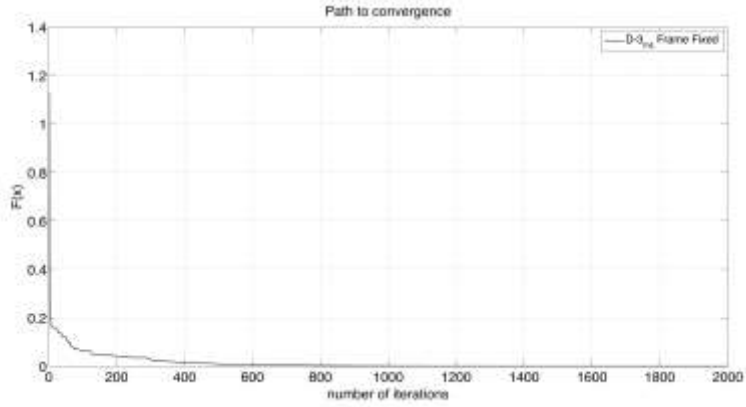


Figure 3.38 Path to convergence for cases *D2-FA-FF* (top) and *D2-FA-FH* (bottom)

If the damage is assigned to both elements 2 and 3 (*D3-FA-FF* and *D3-FA-FH*) the convergence is achieved in 872 seconds after 1197 iterations with fixed elements, while with hinges, the solution in achieved in 155 seconds after 696 iterations, as shown in Figure 3.39.



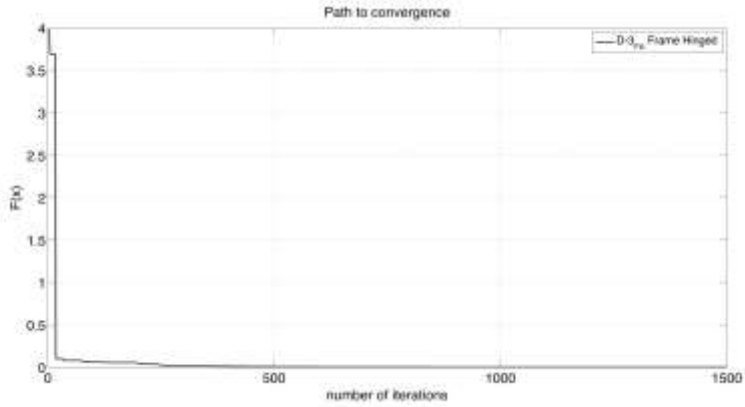


Figure 3.39 Path to convergence for cases *D3-FA-FF* (top) and *D3-FA-FH* (bottom)

When the damage is introduced in a central element, the fifteenth (*D4-FA-FF* and *D4-FA-FH*), a null value of the objective function, as underlined in Figure 3.40, is achieved in 508 seconds after 855 iterations, and in 160 seconds after 641 iterations, respectively.

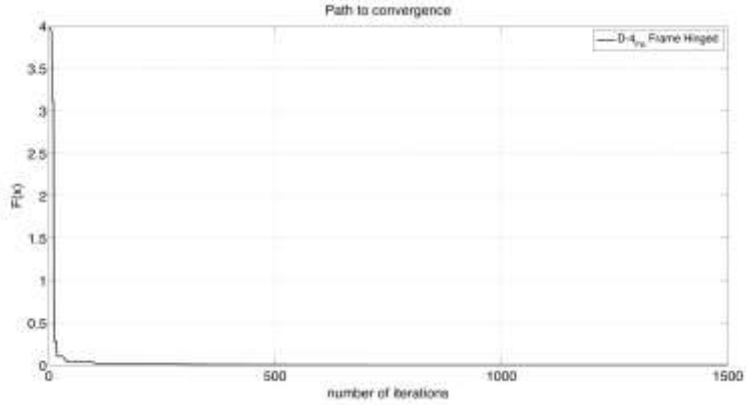
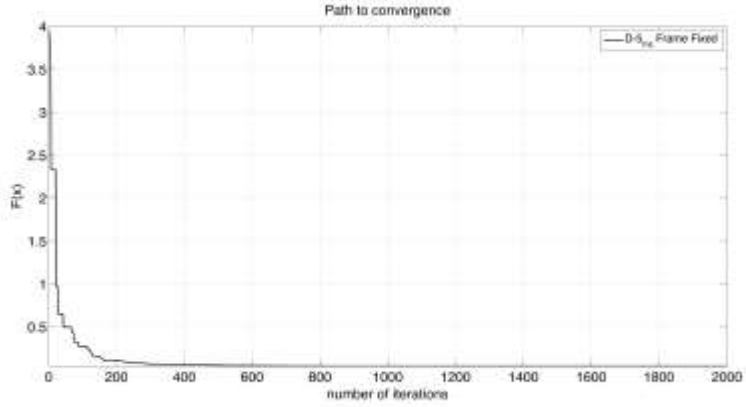


Figure 3.40 Path to convergence for cases *D4-FA-FF* (top) and *D4-FA-FH* (bottom)

Finally, last two cases, namely *D5-FA-FF*, *D5-FA-FH* and *D6-FA-FF*, *D6-FA-FH*, introduce multiple symmetric and non-symmetric damage scenarios. Under these assumptions, for the *D5* cases, the convergence is reached after 1529 iterations in 350 seconds whether the fixed structure is considered, while it is achieved in 115 seconds after 890 iterations whether the hinged structure is considered. Whereas, considering the *D6* cases, the solution is achieved in 371 seconds after 1374 iterations, and in 112 seconds after 766 iterations, respectively.

The paths to convergence for these four analyses are reported in Figure 3.41 and Figure 3.42.



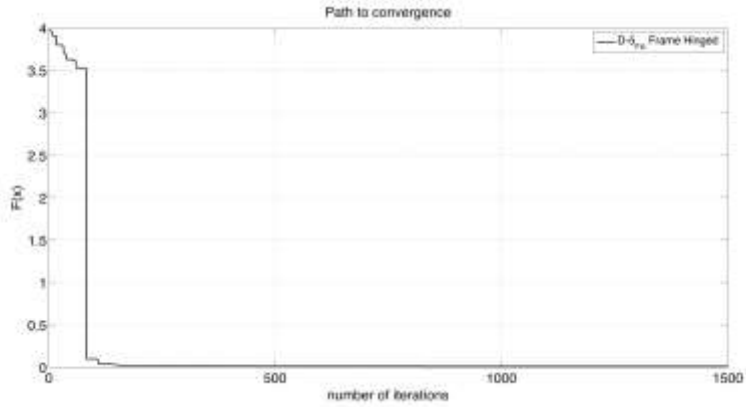


Figure 3.41 Path to convergence for cases *D5-FA-FF* (top) and *D5-FA-FH* (bottom)

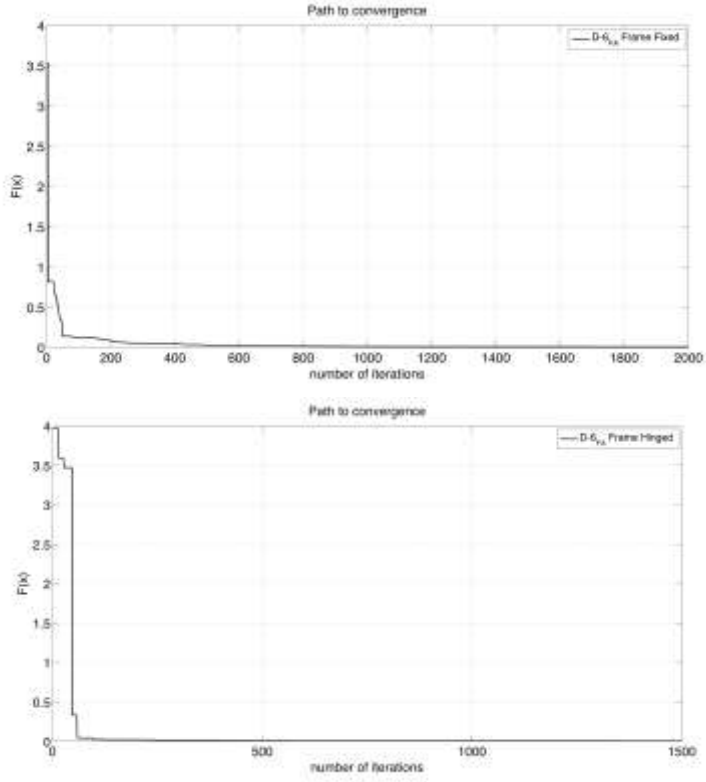


Figure 3.42 Path to convergence for cases *D6-FA-FF* (top) and *D6-FA-FH* (bottom)

3.3.4 Studies on damaged structures via CS

As the previous Sections, the analyses were carried out also with the CS. Again, the structure with fixed elements is initially considered, and then the internal hinges are introduced.

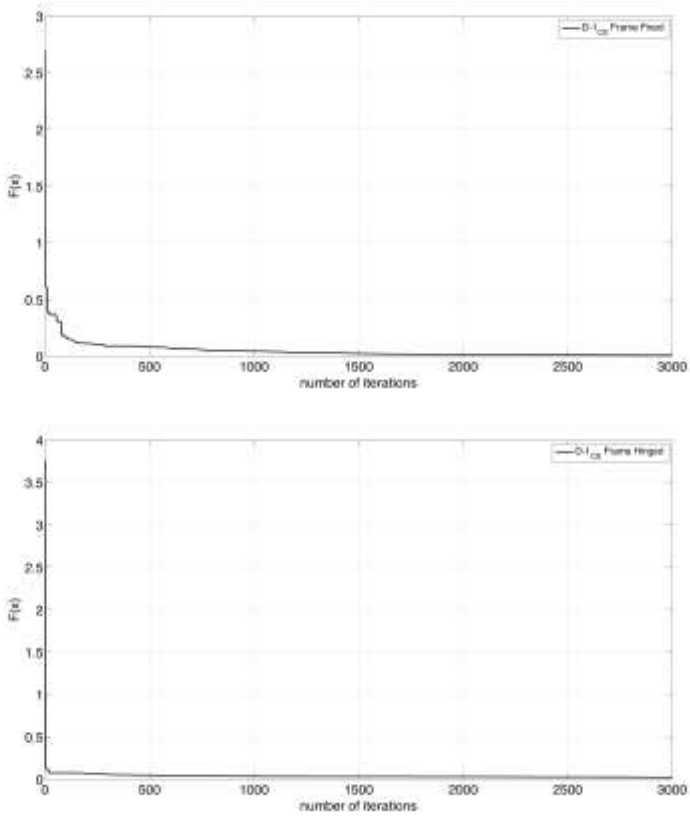


Figure 3.43 Path to convergence for cases *D1-CS-FF* (top) and *D1-CS-FH* (bottom)

The *D1-CS-FF* analysis assigns the damage in the second element and gets the convergence after 2161 iterations in 500 seconds. The same structure, but hinged (*D1-CS-FH*), achieves the null solution in 304 seconds after 2856 iterations. Figure 3.43 shows the paths to convergences for both cases.

The further analysis (*D2-CS-FF* and *D2-CS-FH*) introduces the damage in the third element and it achieves the best solution in 377 seconds after 2010 iterations and in 245

seconds after 2019 iterations, respectively. Figure 3.44 presents both the paths to convergence for the abovementioned analyses.

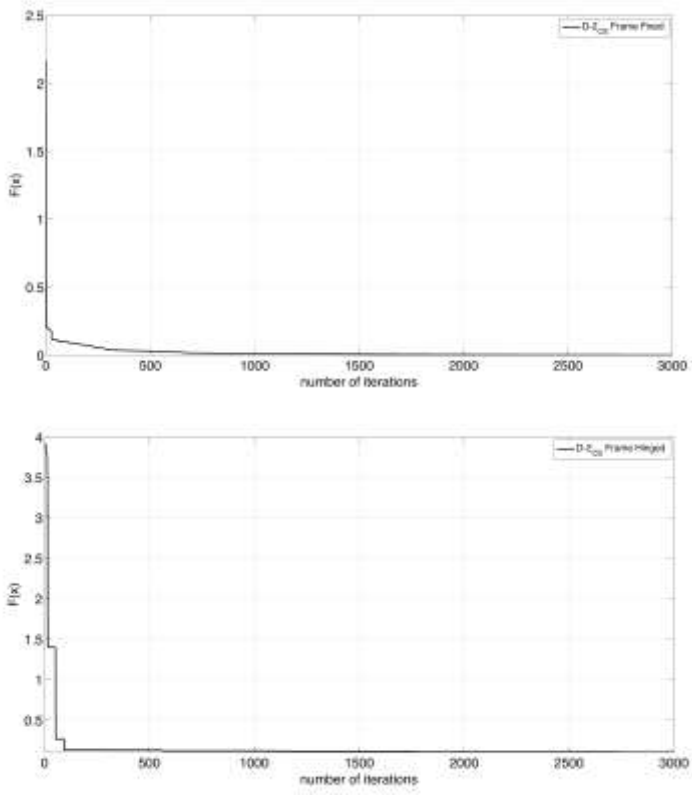


Figure 3.44 Path to convergence for cases *D2-CS-FF* (top) and *D2-CS-FH* (bottom)

Then, the damage is introduced in both elements 2 and 3, labelling the analyses as *D3-CS-FF* and *D3-CS-FH*, and the convergence is reached in 284 seconds after 1928 iterations with fixed elements, and in 459 seconds after 696 iterations, as shown in Figure 3.45.

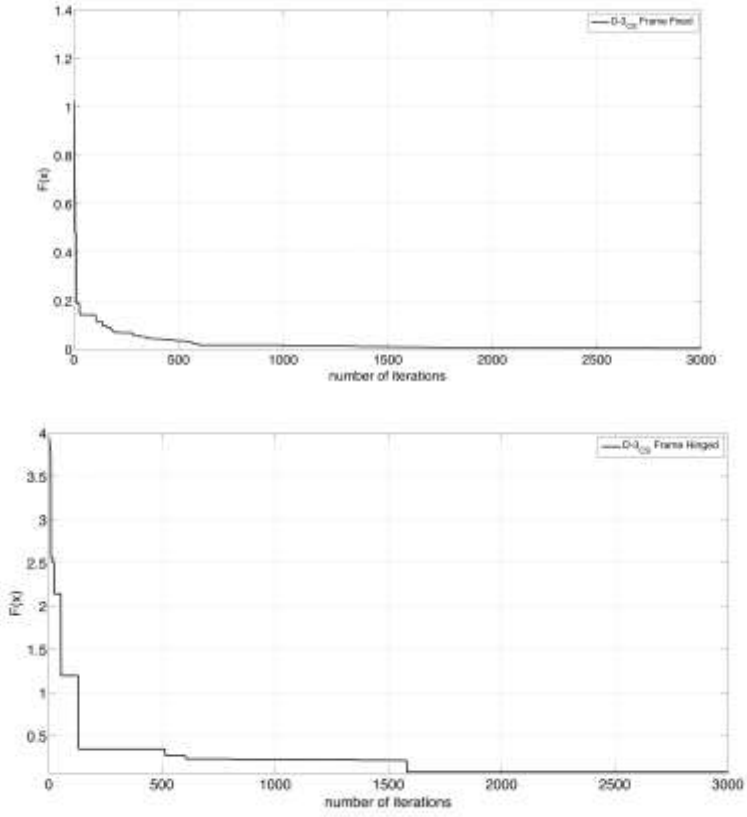


Figure 3.45 Path to convergence for cases *D3-CS-FF* (top) and *D3-CS-FH* (bottom)

When the damage is introduced in a central element, as the number 15, (*D4-CS-FF* and *D4-CS-FH*), the null value of the fitness function, is achieved in 257 seconds after 2351 iterations, and in 379 seconds after 2545 iterations, respectively, as illustrated in Figure 3.46.

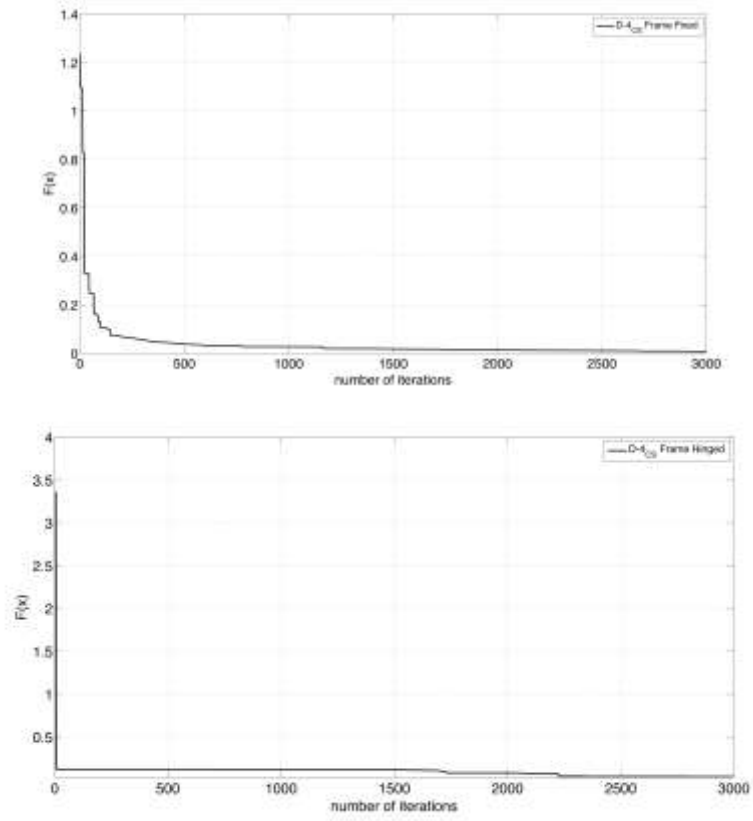


Figure 3.46 Path to convergence for cases *D4-CS-FF* (top) and *D4-CS-FH* (bottom)

Finally, last two cases, labelled as *D5-CS-FF*, *D5-CS-FH* and *D6-CS-FF*, *D6-CS-FH*, introduce multiple symmetric and non-symmetric damage scenarios. So, the convergence is reached after 2698 iterations in 306 seconds considering the fixed structure, whereas it is achieved in 436 seconds after 2512 iterations considering the hinged structure, carrying out the *D5* cases. While, performing the *D6* cases, the solution is achieved in 267 seconds after 2647 iterations, and in 458 seconds after 2134 iterations, respectively.
The paths to convergence for these analyses are illustrated in Figure 3.47 and Figure 3.48, respectively.

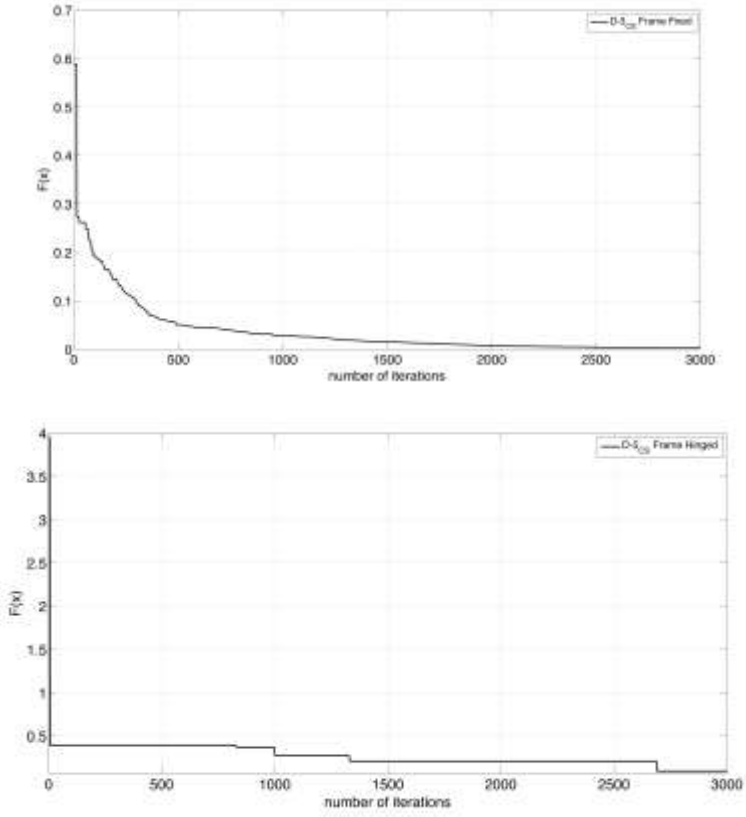
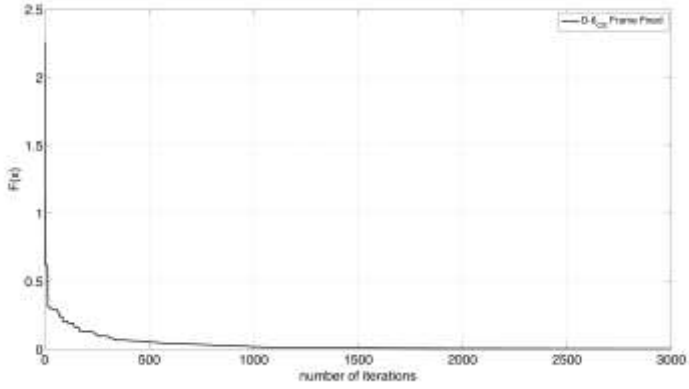


Figure 3.47 Path to convergence for cases *D5-CS-FF* (top) and *D5-CS-FH* (bottom)



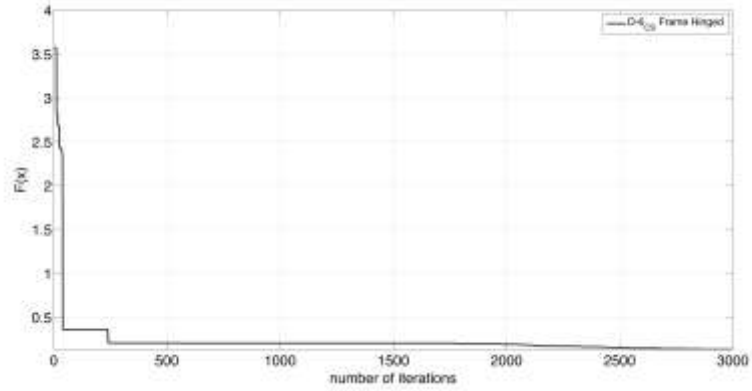
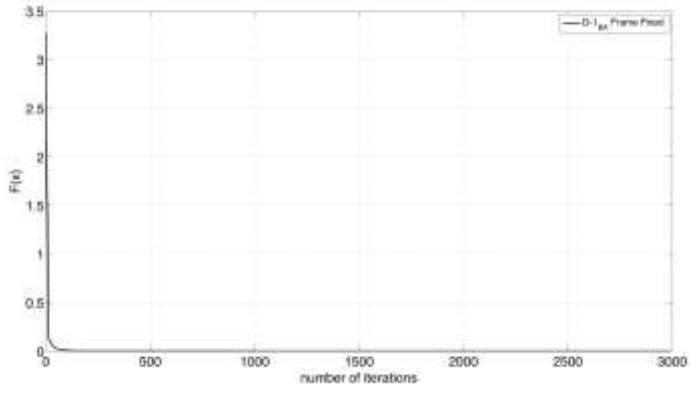


Figure 3.48 Path to convergence for cases *D6-CS-FF* (top) and *D6-CS-FH* (bottom)

3.3.5 Studies on damaged structures via BA

Last set of analyses were carried out also via BA. Thus, the structure with fixed elements is considered, and then the internal hinges are introduced.



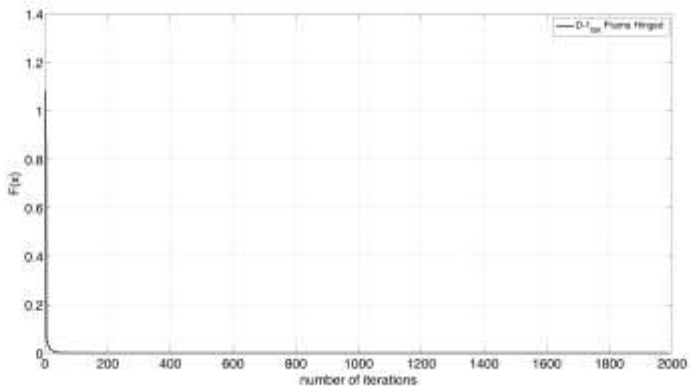
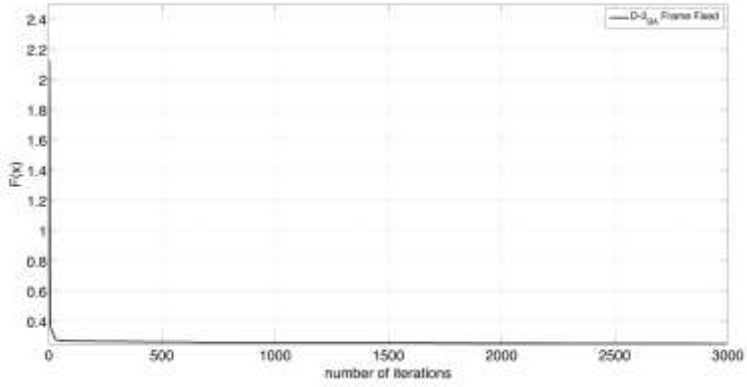


Figure 3.49 Path to convergence for cases *D1-BA-FF* (top) and *D1-BA-FH* (bottom)

The *D1-BA-FF* analysis sets the damage in the element 2 and reaches the convergence after 1203 iterations in 597 seconds. The same structure, but hinged (*D1-BA-FH*), gets the null solution in 247 seconds after 735 iterations. Figure 3.49 shows the paths to convergences for both cases.

The second analysis, namely *D2-BA-FF* and *D2-BA-FH*, assigns the damage in the element 3 and it has the best solution in 561 seconds after 2182 iterations and in 252 seconds after 1398 iterations, respectively. Figure 3.50 illustrates the paths to convergence for the analyses just discussed.



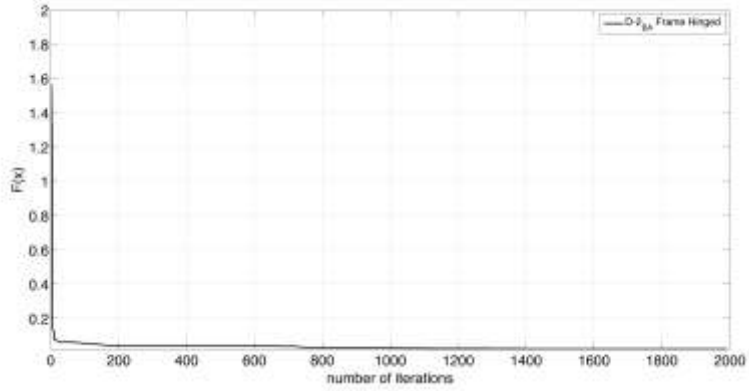
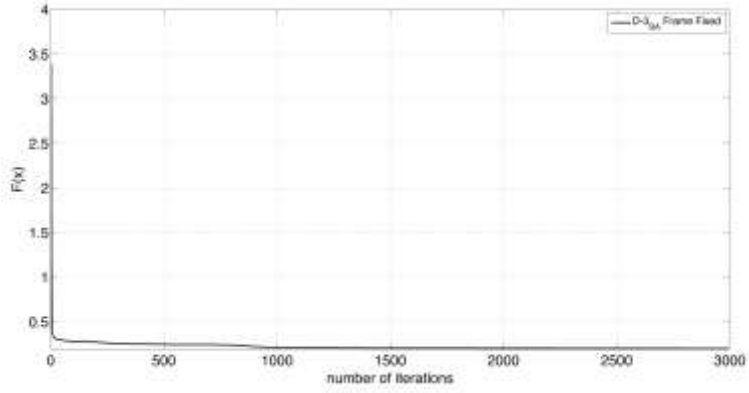


Figure 3.50 Path to convergence for cases *D2-BA-FF* (top) and *D2-BA-FH* (bottom)

Hence, the damage is introduced in elements 2 and 3, (*D3-BA-FF* and *D3-BA-FH*), and it converges in 570 seconds after 1949 iterations with fixed elements, and in 268 seconds after 1114 iterations, as shown in Figure 3.51.



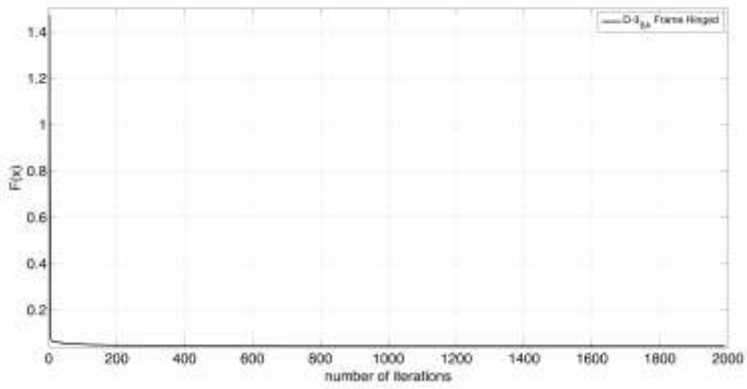
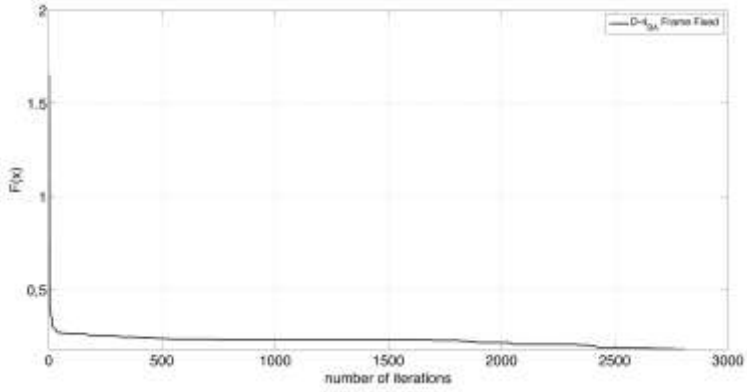


Figure 3.51 Path to convergence for cases *D3-BA-FF* (top) and *D3-BA-FH* (bottom)

When the damage is assigned to a middle element, i.e., the 15th, (*D4-BA-FF* and *D4-BA-FH*), the null value is carried out in 601 seconds after 2651 iterations, and in 287 seconds after 1249 iterations, respectively, as illustrated in Figure 3.52.

At the end, fifth and sixth cases, labelled *D5-BA-FF*, *D5-BA-FH* and *D6-BA-FF*, *D6-BA-FH*, introduce multiple symmetric and non-symmetric damage scenarios. Hence, the null solution is reached after 2019 iterations in 513 seconds considering the fixed structure, while it is achieved in 306 seconds after 1633 iterations considering the hinged structure, carrying out the *D5* cases. Whereas, carrying over the *D6* cases, the solutions are reached in 1158 seconds after 1683 iterations, and in 240 seconds after 1559 iterations, respectively.



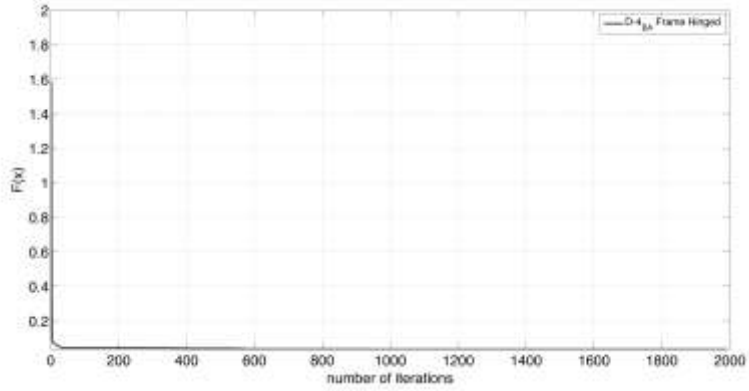
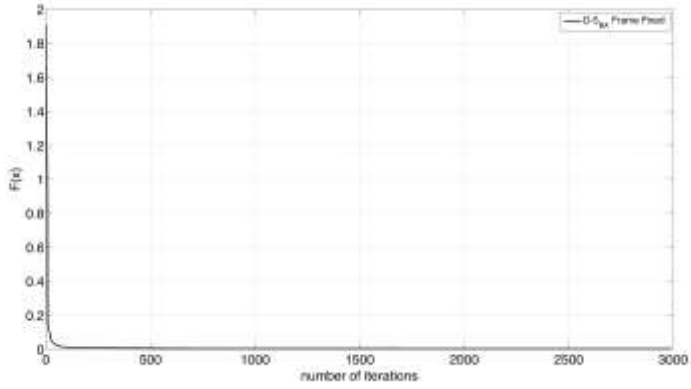


Figure 3.52 Path to convergence for cases *D4-BA-FF* (top) and *D4-BA-FH* (bottom)

The paths to convergence for these analyses are plotted in Figure 3.53 and Figure 3.54, respectively.



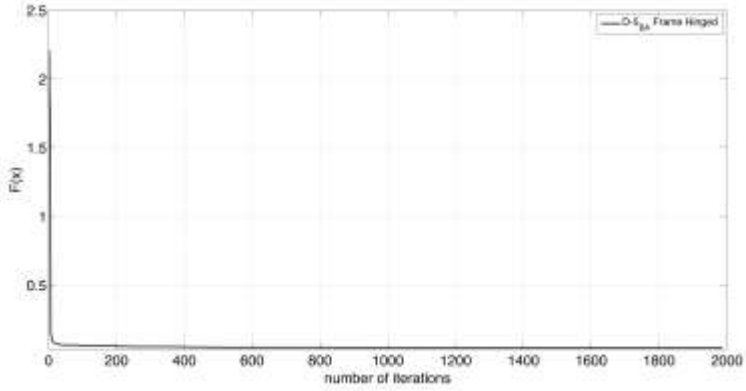


Figure 3.53 Path to convergence for cases *D5-BA-FF* (top) and *D5-BA-FH* (bottom)

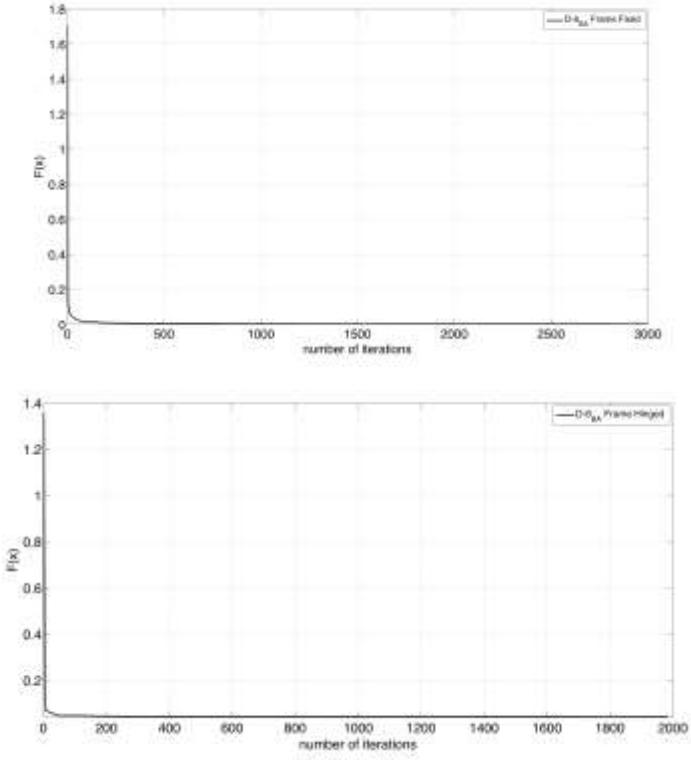


Figure 3.54 Path to convergence for cases *D6-BA-FF* (top) and *D6-BA-FH* (bottom)

3.3.6 Comparison between the methods

After all the analyses were performed, the tools employed for solving the optimization problem under different damage scenarios are compared. In all the fourteen analyses, algorithms allow to converge and to identify the correct scenario within the maximum number of iterations. Also herein, as in Section 3.2.4, the efficiency of the methods is assessed in terms of both the number of iterations and the time duration of the analyses. In order to summarize the results of each performed analysis, with both fixed and hinged structures, the aforementioned performance parameters are plotted in Figures from Figure 3.55 to Figure 3.58.

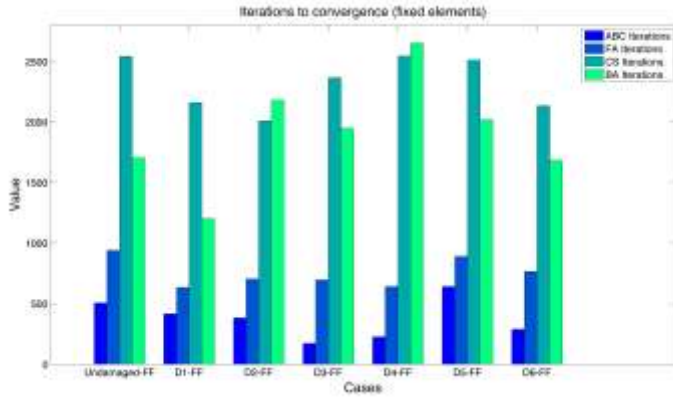


Figure 3.55 Number of iterations to converge (frame structure with fixed elements)

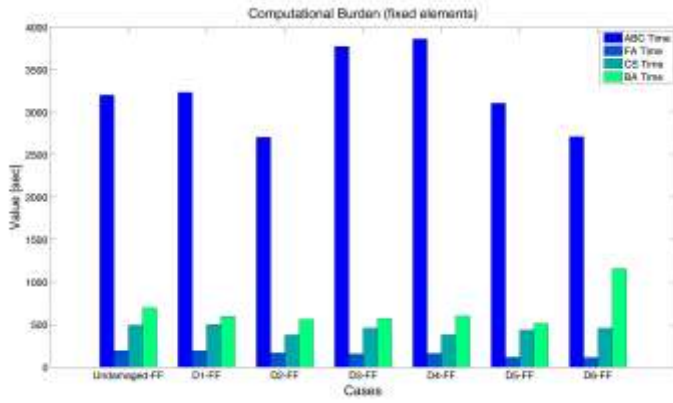


Figure 3.56 Time duration of the analyses (frame structure with fixed elements)

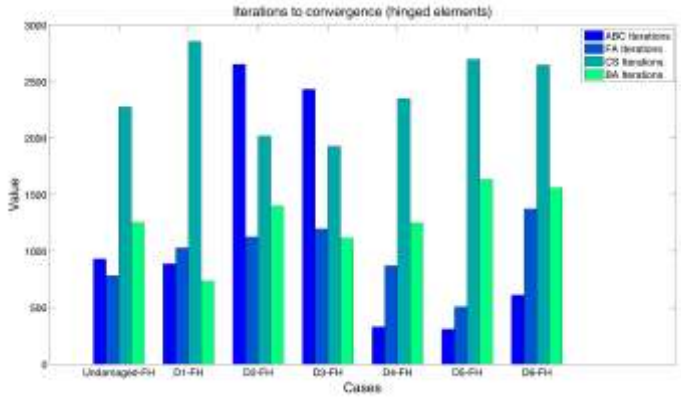


Figure 3.57 Number of iterations to converge (frame structure with hinged elements)

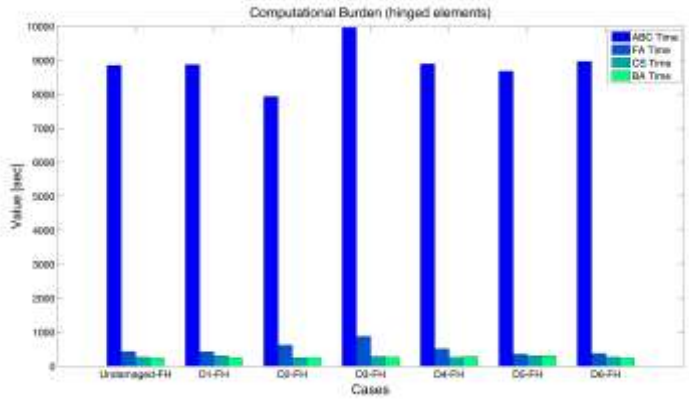


Figure 3.58 Time duration of the analyses (frame structure with hinged elements)

It is shiny that the FA requires, in each analysis, less computational burden than the ABC algorithm and the CS. Indeed, it converges faster even if in some cases it performs a number of iterations higher than the ABC, CS, and BA algorithms. As expected, a reduced number of iterations is needed to handle the problem of decreased dimensionality, but not for the CS where the number has been set same for hinged and fixed structures. Particularly, the performance of the ABC algorithm is significantly affected by the size of the problem.

3.4 The De Gasperi Bridge

The De Gasperi Bridge is a cable-stayed bridge situated in Parma, a city in the north of Italy and can be considered one of the most important infrastructure built in the Country. Figure 3.59 frames the structure in the neighborhood. It is an important structure for the connection between two crucial manufacturing areas [14].

Figure 3.59 Aerial view of the bridge

The geometry of the bridge is asymmetric, with an antenna tilted at 72° over the horizontal, 75m high and 79m long.
The two main spans are 70m and 130m respectively. Figure 3.60 draws the general layout of the bridge taken as case-study. The bridge deck is subdivided into two carriageways 11,4m width, with a 5m gap. Moreover, the bridge deck has mixed steel concrete section. Each of the two parts of the deck is realized with three longitudinal beams, linked by ten concrete girders. Seven of them host the device where the cable-stays are installed [15].
Moreover, 33 stay cables are installed, and 23 of them support the bridge deck. A reinforced concrete antenna is 75m high; and it is mainly composed by two partition walls whose width is variable from 5m to 11m. Furthermore, it is tilted at 18° over the vertical.

Figure 3.60 The De Gasperi Bridge

It is worth noting that on the bridge is arranged a monitoring system which is useful for recording the response of the structure under dynamic loads. Such system is composed by several devices which collect the significant parameters about the behavior of the bridge. Moreover, there exists a data acquisition unit and a transmission data system.
The devices installed on the bridge can be grouped into two different categories: the first one is dedicated to the measurement of the deformation of the deck and the antenna, while the second one is dedicated to the ambient vibration measurements.
The devices, employed to the measurement of the deformable parts, record the displacements of the deck as to the antenna and to the supports, the deformation of the base of the antenna and the deck, and the dynamic response of the deck.
For this purpose, six transducers that measure the relative displacements between the antenna and the deck and between the antenna and the deck compared to the supports, sixteen extensometers and a tri-axial accelerometer are utilized.

3.4.1 The finite element analyses

The bridge is modeled with the finite element method. It is designed by a series of beam elements that compose the bridge and reply the same behavior of the real structure. A detailed model were performed in order to frame any possible issue in the proper manner. The design of the structure is carried out by the finite element software, namely MSC® Marc Mentat [2].
An option provided by Marc Mentat is the possibility to extract and then save matrices into single files. Thus, even the model has a large complexity in the pre-processor of Marc

Mentat, the resulting in the assemblage of matrices can be exported into another environment, e.g. MATLAB® [3], to follow an optimization process.

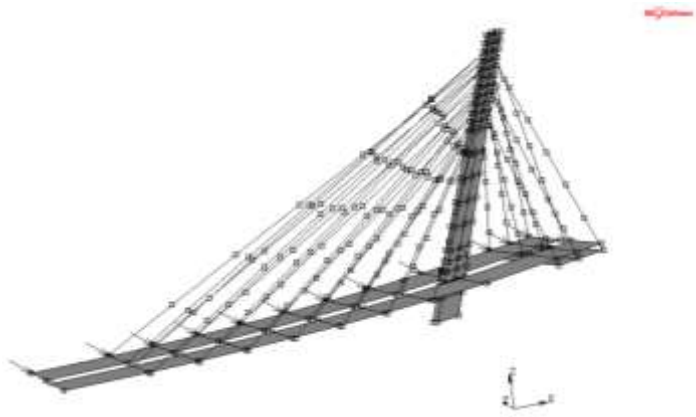


Figure 3.61 Numerical model of the De Gasperi Bridge in Marc Mentat environment

In Figure 3.61, the finite element model built in Marc Mentat environment is shown. The structural configuration is conceived with 471 elements and 373 nodes. Each element is discretized as a beam or a shell, then it becomes easy calculating the number of degrees of freedom, by multiplying the number of nodes by six (i.e., the number of freedom of each element), and obtaining 2238. The bridge is simply supported on the edges. In the following, this configuration will be denoted as "Undamaged".

Table 3.20 Features of the De Gasperi Bridge

Property	Value
Length (l_{el})	177m
Number of nodes per element (n_e)	2/4
Number of elements (m)	471
Number of nodes (n)	373
Number of degrees of freedom	2238
Young modulus (E)	2.0×10^{11} N/m^2 (steel)/ 3.0×10^{8} N/m^2 (RC)
Poisson ratio (v)	0.2 (steel)/0.3 (RC)
Mass density (ρ)	7.8×10^{7} kg/m^3 (steel)/ 2.5×10^{7} kg/m^3 (RC)

The material is mixed steel and reinforced concrete with a Young modulus, a mass density, and a Poisson defined as in Table 3.20. The length of each element varies.

Moreover, the damage is introduced as a degradation of a stiffness. Hence, it is denoted by multiplying the stiffness matrix associated to the whole structure, by a non-dimensional quantity ψ, whose value falls in the interval from 0 to 1.
So, one can express the degradation of the stiffness as:

$$\mathbf{K}_{dam} = \sum_{i=1}^{m} \psi_i \mathbf{K}_{dam,i} \qquad (43)$$

where $\psi_i = [0,1]$. When such value is equal to 1, it means that the structure is in undamaged condition. Each different damage scenario herein considered is outlined in Table 3.21.

Table 3.21 Different structural configurations and damage scenarios (De Gasperi Bridge)

Structural configuration	Damaged element(s)	ψ_i
Undamaged	-	1
Damage 1 (D1)	38	0.7
Damage 2 (D2)	106	0.7
Damage 3 (D3)	151	0.4
Damage 4 (D4)	322÷327	0.7
Damage 5 (D5)	29 – 64	0.7 – 0.7
Damage 6 (D5)	29 – 64	0.5 – 0.7

Thus, the exact values of natural frequencies and mode shapes are achieved by solving the eigenproblem stated in Equation (27) for all the structural configurations proposed in Table 3.22.

Table 3.22 First exact nine modal frequencies for different values of element stiffness coefficient (De Gasperi Bridge)

Structural configuration	Exact values of first 9 circular frequencies (rad/s)								
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
Undamaged	5,600	5,135	5,141	5,136	5,135	5,135	5,329	5,600	5,135
Damage 1	11,892	10,151	11,791	11,791	11,788	11,787	11,787	11,892	10,151
Damage 2	13,668	13,218	13,473	13,472	13,471	13,472	13,471	13,668	13,218
Damage 3	14,122	14,018	14,019	14,018	14,018	14,721	14,771	14,122	14,018
Damage 4	16,776	16,518	16,521	16,520	16,530	16,527	16,527	16,776	16,518
Damage 5	16,801	16,615	16,646	16,578	16,580	16,580	16,580	16,801	16,615
Damage 6	21,539	21,287	21,120	21,074	21,094	21,096	21,096	21,539	21,287

The quantities $\bar{\boldsymbol{\omega}}_{exact}$ and $\bar{\boldsymbol{\Phi}}_{exact}$ have size $n_{dof} \times 1$ and $n_{dof} \times n_{dof}$ respectively. In each analysis, both those quantities are used as input parameters of the solving optimization

technique. Then the weight function, *w*, is calibrated with a value equal to 1, in order to account the eigenvectors' contribution.
Consequently the identification of the modal features, they are introduced in the objective function of Eq. (32) and the tools are applied to deal with the optimization problem stated in Eq. (36). Each analysis wants to recognize the damage scenario previously defined.
The existence of the domain of each design variable is included in the defined interval that was properly framed by a preliminary study on the undamaged structure. Such an important operation because it is useful for the convergence and for the calibration of the control parameters of each algorithm. After this procedure, the damage detection and localization is pursued for each case in Table 3.21.

3.4.2 Preliminary analyses on undamaged structure

As first operation, the control parameters for each algorithm have to be set. Tables from Table 3.23 to Table 3.26 show it.

Table 3.23 Control parameters of ABC (De Gasperi Bridge)

Control parameters of ABC	**Values for each structural configuration**					
	D_{ABC}-1	**D_{ABC}-2**	**D_{ABC}-3**	**D_{ABC}-4**	**D_{ABC}-5**	**D_{ABC}-6**
CS, size of the initial population of honeybees	50	50	50	50	50	50
C_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
λ, limit number for abandoning the food search	100	100	100	100	100	100

Table 3.24 Control parameters of FA (De Gasperi Bridge)

Control parameters of FA	**Values for each structural configuration**					
	D_{FA}-1	**D_{FA}-2**	**D_{FA}-3**	**D_{FA}-4**	**D_{FA}-5**	**D_{FA}-6**
NP, size of the initial population of fireflies	100	100	100	100	100	100
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
ζ, randomization number	0.5	0.5	0.5	0.5	0.5	0.5
β_{min}, minimum attractiveness	0.2	0.2	0.2	0.2	0.2	0.2
γ, absorption coefficient	1.0	1.0	1.0	1.0	1.0	1.0

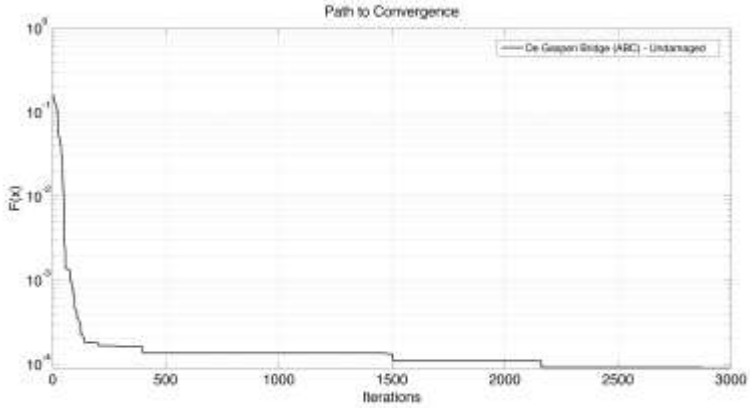
Table 3.25 Control parameters of CS (De Gasperi Bridge)

Control parameters of CS	Values for each structural configuration					
	D_{ABC}-1	D_{ABC}-2	D_{ABC}-3	D_{ABC}-4	D_{ABC}-5	D_{ABC}-6
n, number of nests	100	100	100	100	100	100
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
p_a, discovery rate	100	100	100	100	100	100

Table 3.26 Control parameters of BA (De Gasperi Bridge)

Control parameters of BA	Values for each structural configuration					
	D_{ABC}-1	D_{ABC}-2	D_{ABC}-3	D_{ABC}-4	D_{ABC}-5	D_{ABC}-6
n, size of the initial population of bats	100	100	100	100	100	100
I_{max}, maximum number of iterations	3000	3000	3000	3000	3000	3000
pulse rate	1	1	1	1	1	1
loudness	0.8	0.8	0.8	0.8	0.8	0.8

The search domain is set in the interval from 0 to 1 in order to frame the solution.
According these assumptions, the analyses labelled as *Undamaged De Gasperi Bridge* are performed and the convergence, is achieved in 407214 seconds after 2162 iterations by ABC, in 160315 seconds after 1434 iterations by FA, in 148950 seconds after 1031 iterations by CS, and in 259117 seconds in 2040 iterations by BA.
Figure 3.62 shows the path to convergence for the undamaged cases.



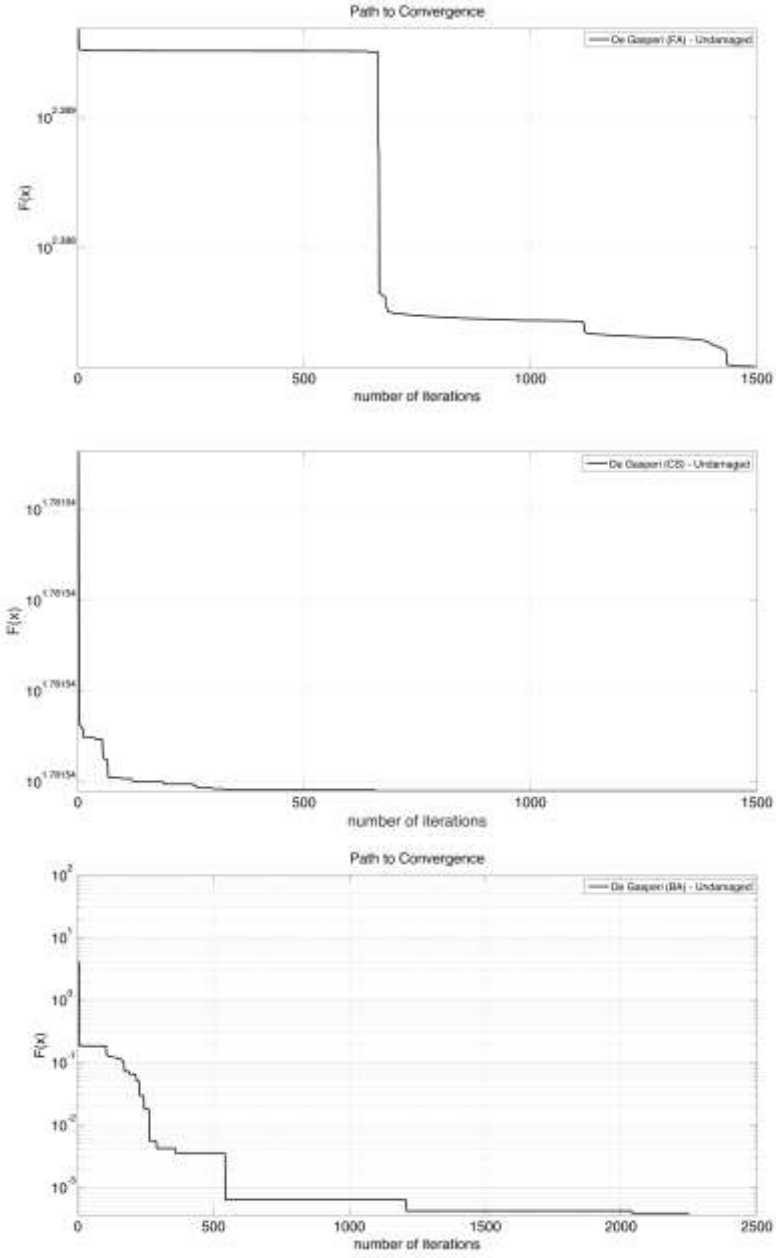


Figure 3.62 Path to convergence for *Undamaged-ABC De Gasperi Bridge* (first), *Undamaged-FA De Gasperi Bridge* (second), *Undamaged-CS De Gasperi Bridge* (third), and *Undamaged-BA De Gasperi Bridge* (fourth)

For sake of completeness, all the analyses carried out on a Mac OsX notebook, 64-bit, 2.8GHz Intel® Core i7 processor with 8GB ram.

3.4.3 Studies on damaged structures via ABC

Six further analyses, as reported in Table 3.21, are performed by applying the artificial bee colony algorithm, for localizing the damage in the structure under different scenarios. The first analysis is focused on element 38, close to the left support, and the damage intensity is equal to 0.7 (*D1-ABC-DGB*). For this purpose, the convergence is reached in 2690 iterations after 472957 seconds and it is shown in Figure 3.63.

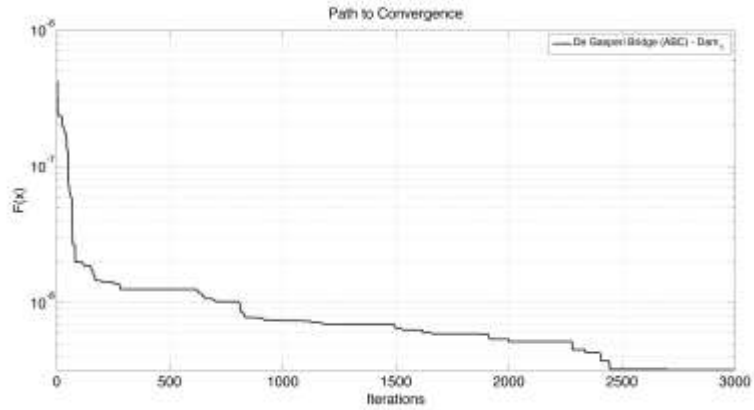


Figure 3.63 Path to convergence for case *D1-ABC-DGB*

Then, the second analysis is performed: it is labelled as *D2-ABC-DGB* and aims to detect the damage in element 106, i.e. an element of the antenna. The damage intensity is the same as the previous case, and the null value of the fitness function is reached in 2853 iterations after 474966 seconds. The path to convergence is shown in Figure 3.64.

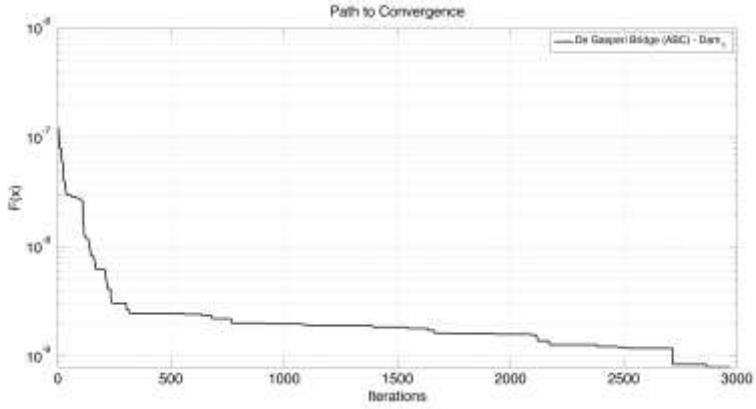


Figure 3.64 Path to convergence for case *D2-ABC-DGB*

The third case (*D3-ABC-DGB*) wants to verify the behavior of an element on the left span, namely element 151, with a damage coefficient equal to 0.4. The path to convergence in Figure 3.65 shows that the null value is achieved in 1668 iterations after 414408 seconds.

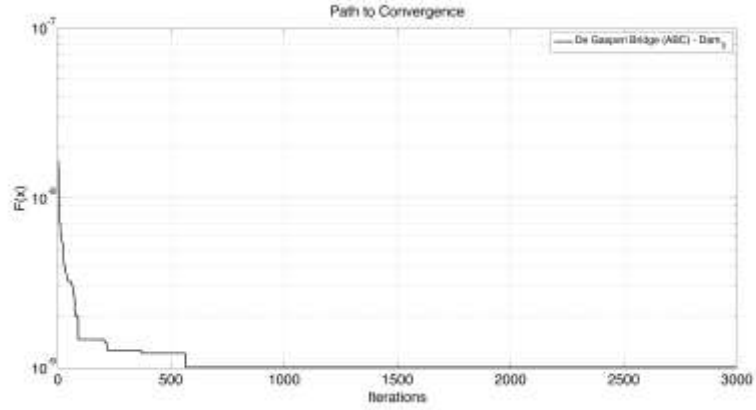


Figure 3.65 Path to convergence for case *D3-ABC-DGB*

When the damage occurs in a stay-cable, the fourth analysis (*D4-ABC-DGB*) is performed. The elements are from the number 322 to 327 with an intensity equal to 0.7. The convergence is reached in 1480 iterations after 47990 seconds, as illustrated in Figure 3.66.

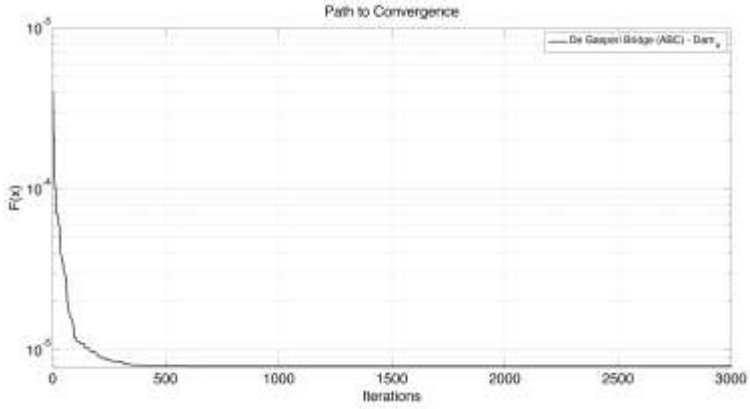
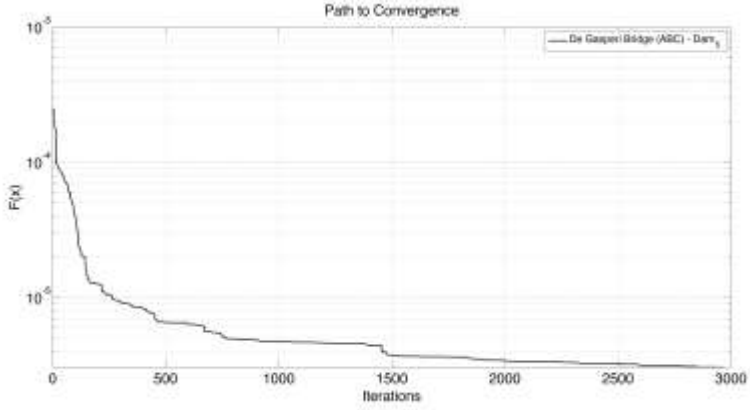


Figure 3.66 Path to convergence for case *D4-ABC-DGB*

Last two cases (*D5-ABC-DGB* and *D6-ABC-DGB*) focus the damage in the same elements (29 and 64, i.e. close to the antenna), but having symmetric (0.7) and non-symmetric (0.5 and 0.7) intensity, respectively. For both cases, the convergence is reached in 2716 iterations after 479961 seconds, and 2669 iterations and 486153 seconds, respectively. Figure 3.67 shows both paths to convergence.



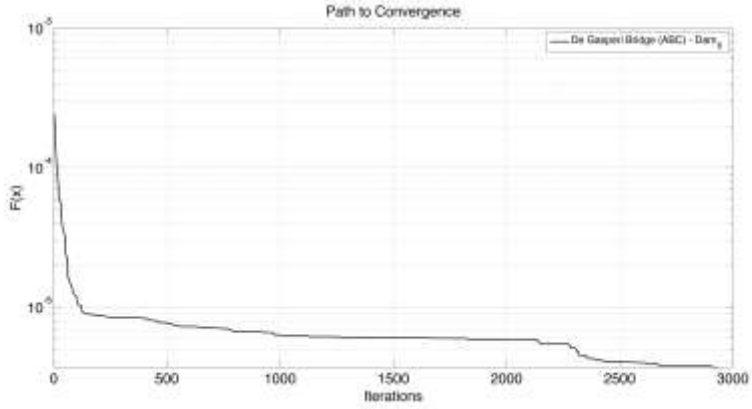


Figure 3.67 Path to convergence for cases *D5-ABC-DGB* (top) and *D6-ABC-DGB* (bottom)

3.4.4 Studies on damaged structures via FA

As for other examples, the same analyses carried out in Section 3.3.2 are herein performed via FA. The first analysis (*D1-FA-DGB*) aims to detect the damage localization in element 38, and the convergence is reached in 1809 iterations after 342155 seconds. Its path to convergence is shown in Figure 3.68.

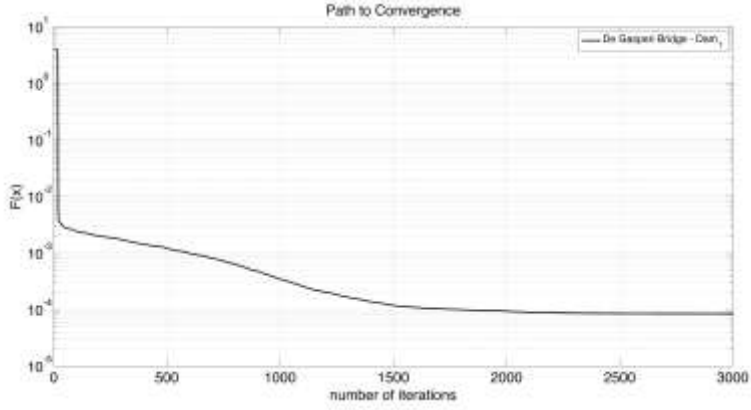


Figure 3.68 Path to convergence for case *D1-FA-DGB*

Then, the *D2-FA-DGB* analysis is carried out: the intensity of the damage is set with 0.7 in an element that composes the antenna and the results of the null solution are hereinafter presented: 2868 iterations, and 348216 seconds, as illustrated in Figure 3.69.

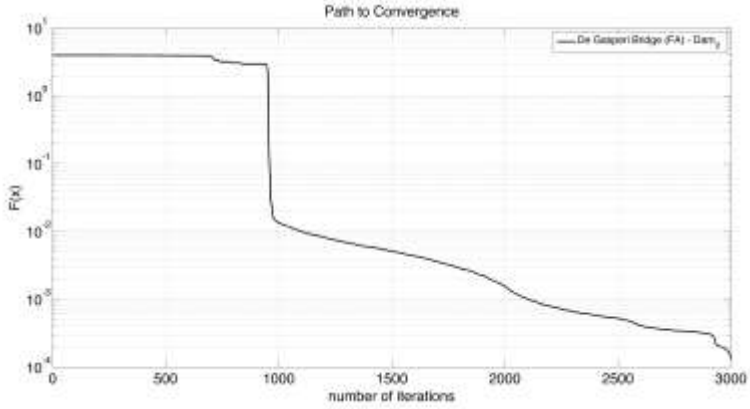


Figure 3.69 Path to convergence for case *D2-FA-DGB*

Considering the third case (*D3-FA-DGB*) a larger intensity of damage is assigned to element 151 for verifying whether the behavior in the middle of the left span is the same of the edge elements. Hence, Figure 3.70 illustrates the path to convergence. The convergence itself is reached in 1955 iterations after 344725 seconds.

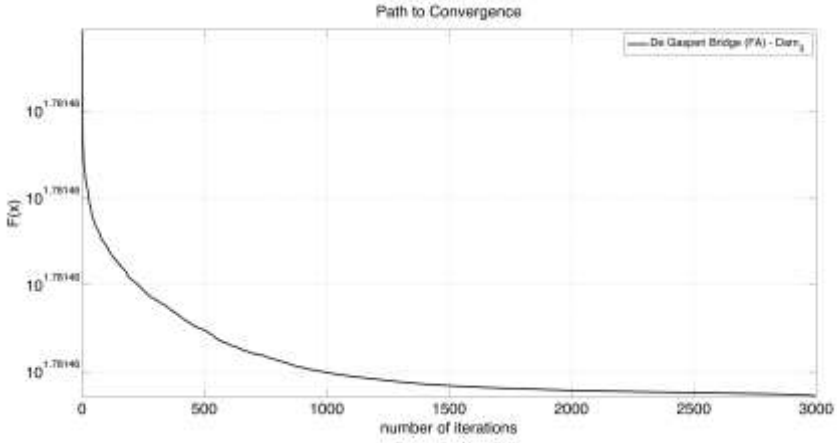


Figure 3.70 Path to convergence for case *D3-FA-DGB*

Fourth analysis (*D4-FA-DGB*) is shaped on a cable-stay, i.e. elements from 322 to 327, and the damage is characterized by a value equal to 0.7. Here, the null value of the objective function is reached in 343996 seconds after 1861 iterations. Even for this case, the path to convergence is shown in Figure 3.71.

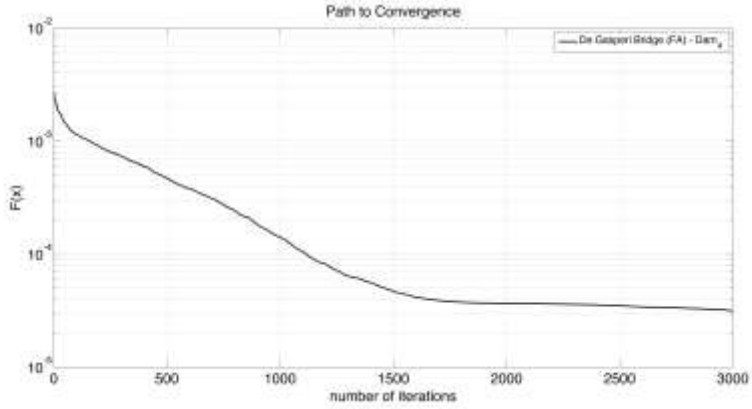
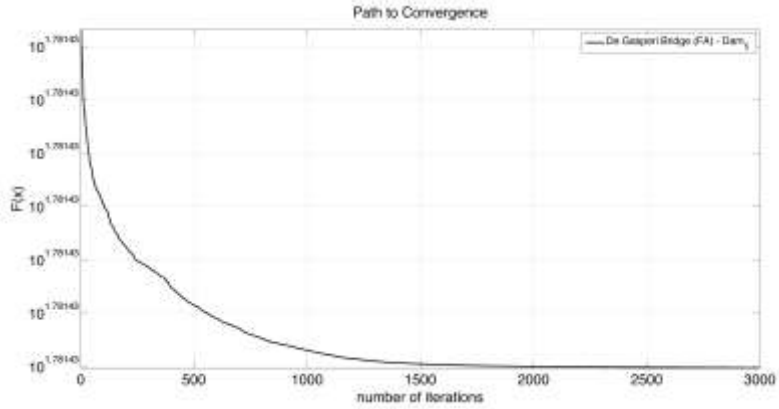


Figure 3.71 Path to convergence for case *D4-FA-DGB*

Last two analyzed cases assign the damage in elements 29 and 64, namely close to the antenna, with intensity 0.7 and 0.7, and then 0.5 and 0.7. Thus, the convergence is achieved in 330142 seconds after 1874 iterations, and in 306106 seconds after 1959 iterations, respectively.



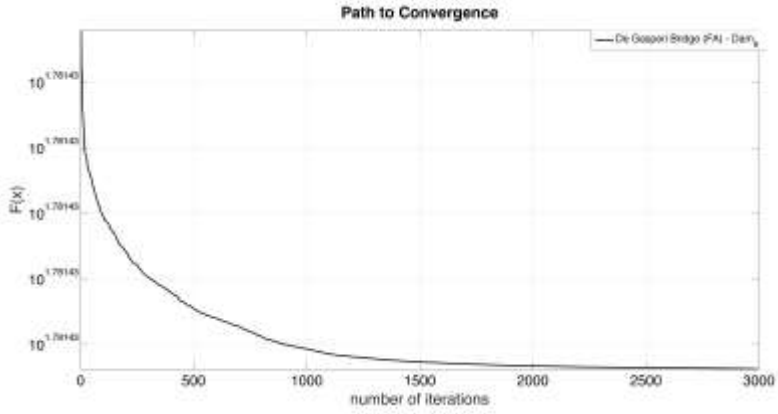


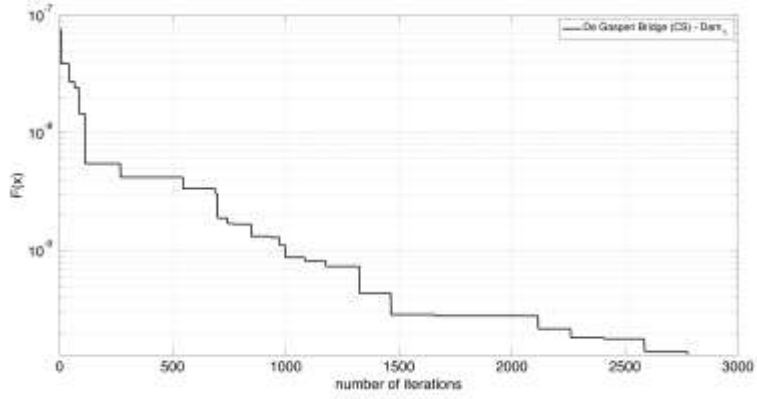
Figure 3.72 Path to convergence for cases *D5-FA-DGB* (top) and *D6-FA-DGB* (bottom)

Figure 3.72 shows both the *D5-FA-DGB* and the *D6-FA-DGB* paths to convergence.

3.4.5 Studies on damaged structures via CS

Once again, the same set of six analyses is performed via cuckoo search for the detection of the damage in the bridge.

The analyzed cases start from the first one (*D1-CS-DGB*) that assigns the damage to the 38th element and gets the convergence in 342100 seconds after 2775 iterations. Then, carrying over the analysis, the second one, namely *D2-CS-DGB*, gives the result in 346219 seconds after 1980 iterations. The paths to convergence are illustrated in Figure 3.73.



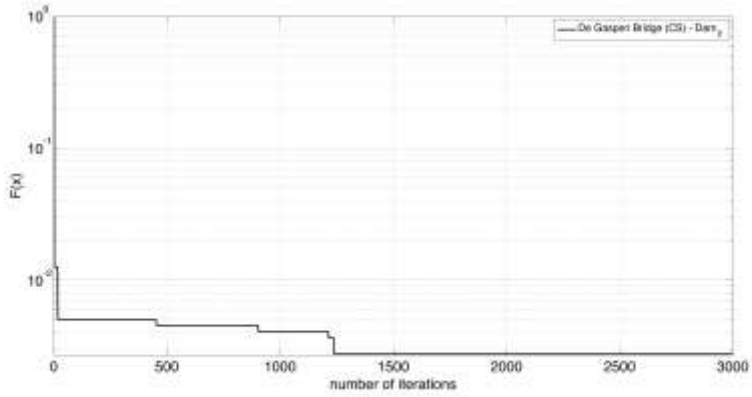


Figure 3.73 Path to convergence for cases *D1-CS-DGB* (top) and *D2-CS-DGB* (bottom)

The following analysis (*D3-CS-DGB*) concerns an element on the left span that is damaged with an intensity equal to 0.4; for such case, the convergence is reached after 2810 iterations in 396281 seconds. The regarding path to convergence is shown in Figure 3.74.

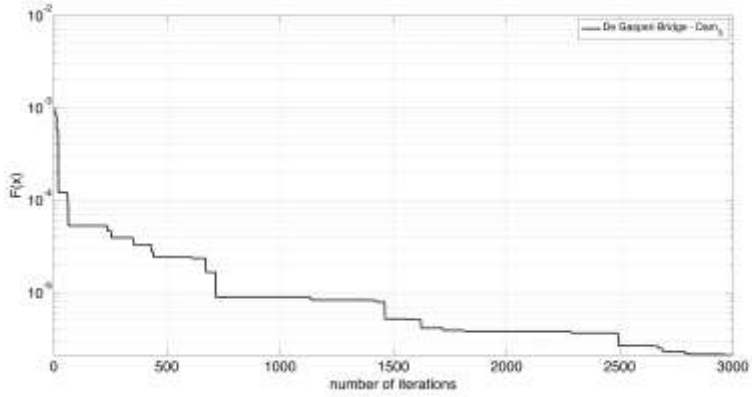


Figure 3.74 Path to convergence for case *D3-CS-DGB*

Hence, a cable stayed is damaged in case *D4-CS-DGB*, and so the value of a null function is achieved in 356554 seconds after 2642 iterations, as highlighted in Figure 3.75.

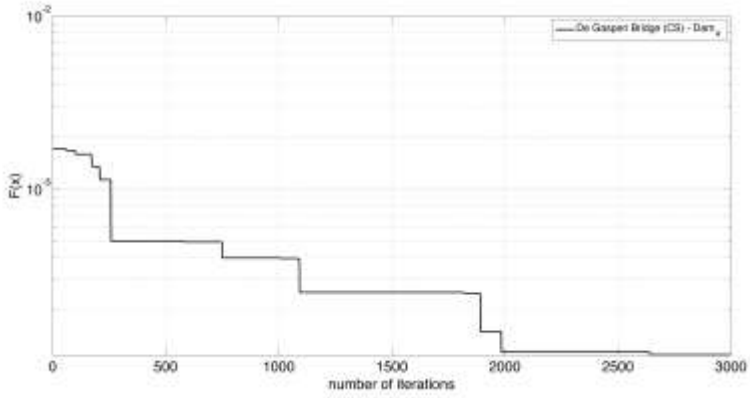
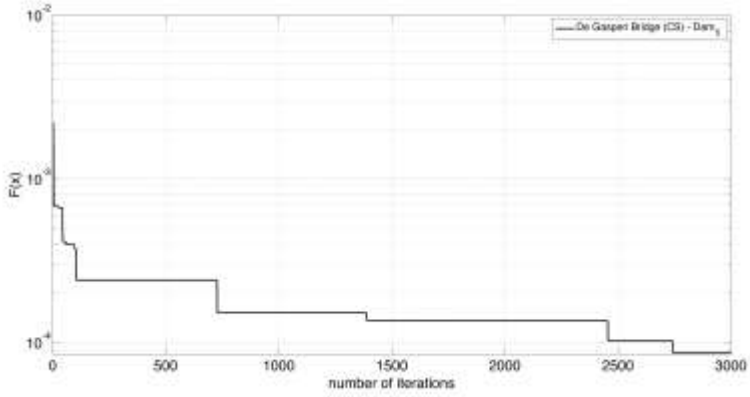


Figure 3.75 Path to convergence for case *D4-CS-DGB*

Finally, last two cases deal with damage assigned in symmetric (*D5-CS-DGB*) and non-symmetric (*D6-CS-DGB*) intensity to elements next to the antenna. The path to convergence in Figure 3.76 shows that it is reached in 387125 seconds after 2742 iterations and in 401712 seconds after 2082 iterations, respectively.



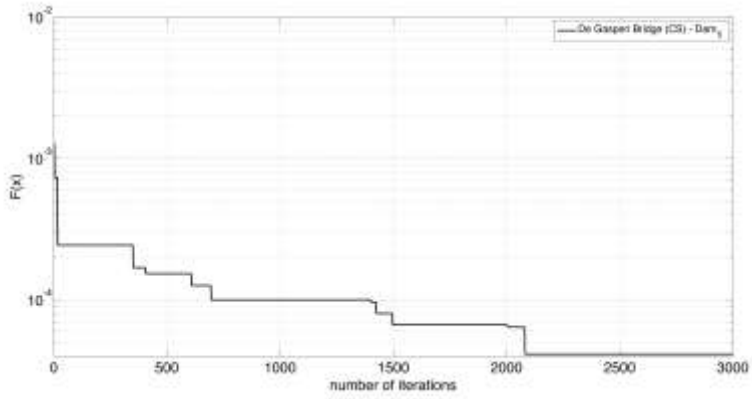


Figure 3.76 Path to convergence for cases *D5-CS-DGB* (top) and *D6-CS-DGB* (bottom)

3.4.6 Studies on damaged structures via BA

Last set of analyses is performed via bat algorithm in order to give a bigger scenario for the comparison between the methods.
The analyses are the same dealt with the previous sections and are replayed again. The first analysis, labelled *D1-BA-DGB* as in Figure 3.77, gives the convergence in 266129 seconds after 2512 iterations.

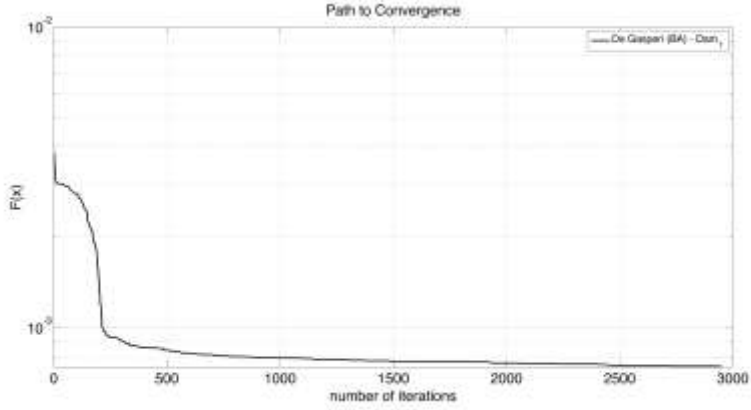


Figure 3.77 Path to convergence for case *D1-BA-DGB*

Second analysis (*D2-BA-DGB*), which focus the research of the damage in element 106 (i.e., antenna), achieve the best solution after 1810 iterations in 265000 seconds, just as in *D3-BA-DGB* where the best optimum is found after 2937 iterations in 296321 seconds

(damage in the left mid-span). Following Figure 3.78 and Figure 3.79 show the path to convergence to both cases.

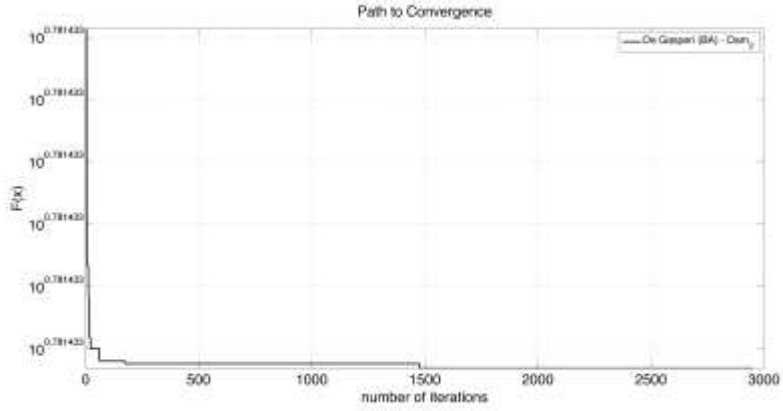


Figure 3.78 Path to convergence for case *D2-BA-DGB*

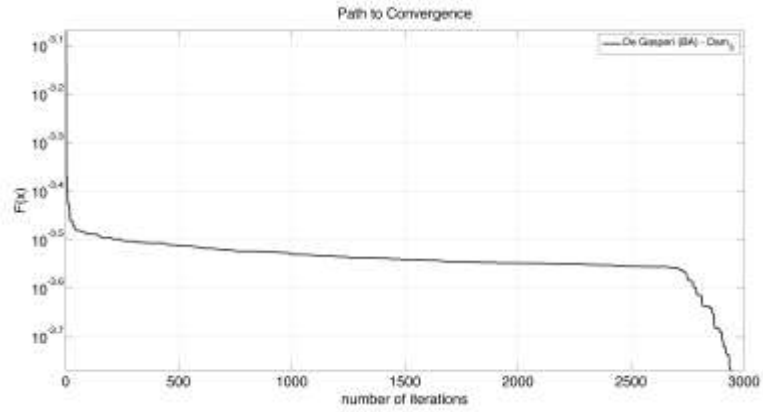


Figure 3.79 Path to convergence for case *D3-BA-DGB*

When introducing the damage in the stay-cable (*D4-BA-DGB*) the solution is achieved in 289226 seconds after 2288 iterations, as the plot in Figure 3.80 illustrates.

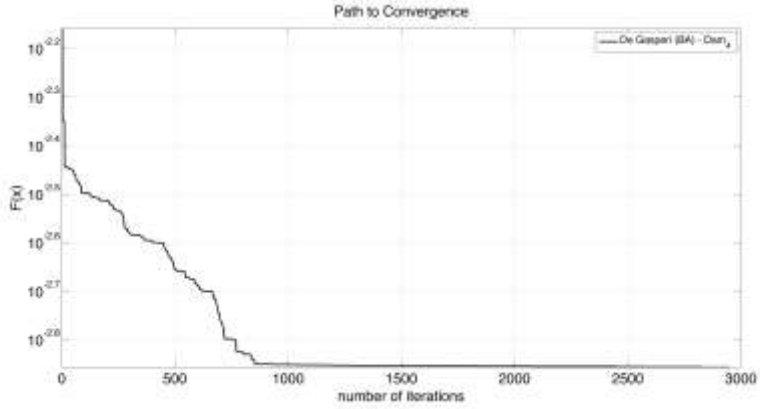
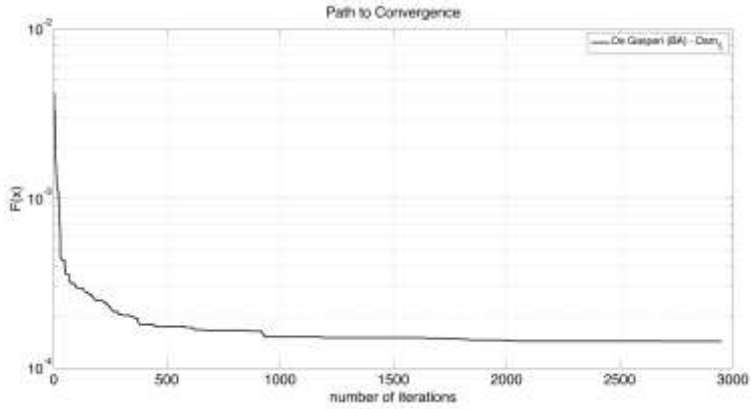


Figure 3.80 Path to convergence for case *D4-BA-DGB*

The fifth case (*D5-BA-DGB*), as the sixth (*D6-BA-DGB*) too, are centered on the damage in elements close to the antenna. For reaching the best solution, the iterations are 2041 and 2470, while the seconds are 294112 and 295009, respectively. Last plots in Figure 3.81 show it.



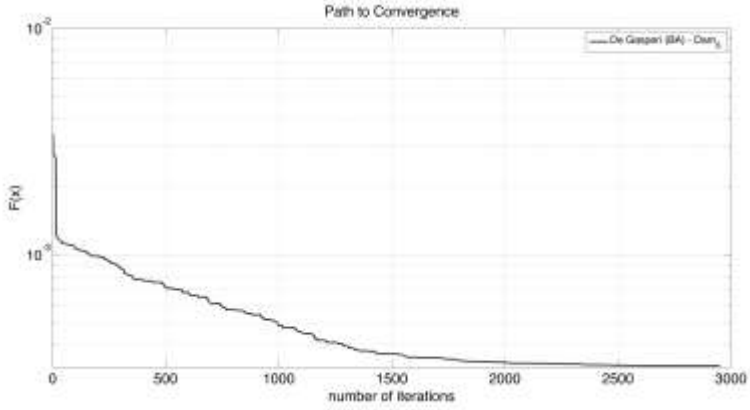


Figure 3.81 Path to convergence for cases *D5-BA-DGB* (top) and *D6-BA-DGB* (bottom)

3.4.7 Comparison between the methods

At the end of the operations of analysis performing, the set of employed tools for solving the optimization issue are compared. In each analysis, algorithms can converge and thus identify the correct scenario within the maximum number of iterations. Hence, the efficiency of each single method is evaluated in terms of both computational burden and number of iterations to converge. Finally, for summarizing the results for each proposed scenario, the performance parameters are illustrated in Figure 3.82 and Figure 3.83.

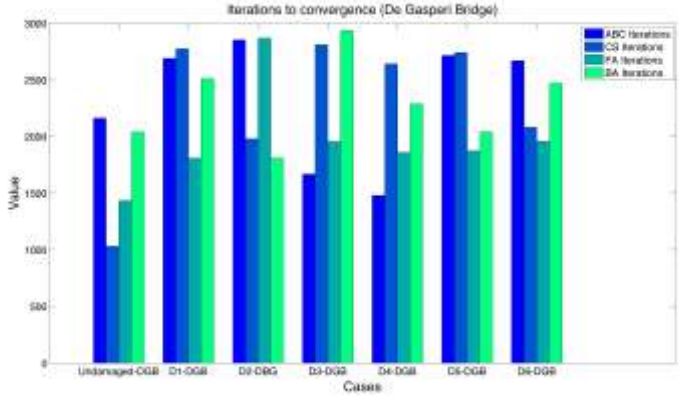


Figure 3.82 Number of iterations to converge (De Gasperi Bridge)

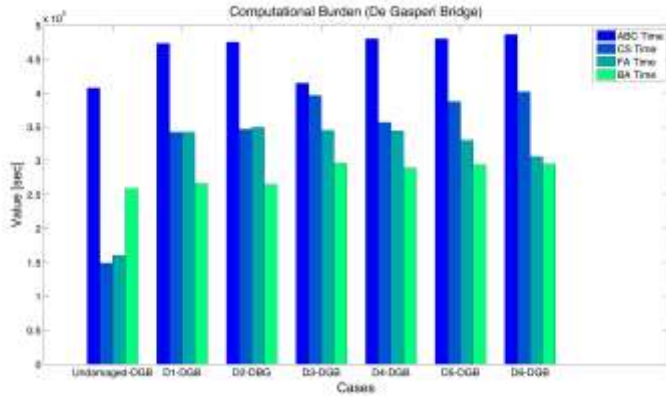


Figure 3.83 Time duration of the analyses (De Gasperi Bridge)

It is shiny that both the firefly algorithm and bat algorithm are more efficient in terms of the duration time of any analysis. It is worth noticing that the number of iterations to convergence for the FA is smaller than the other tool. The performance of the CS are quite satisfactory in terms of computational burden and quite comparable with the FA and BA ones. The ABC algorithm is a strong method, but it is quite slow despite to other adopted tools.

3.5 References

[1] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals – 6th edition*, Ed. Butterworth-Heinemann, 2005.

[2] MSC, *Marc Mentat User Manual*, MSC Software Corp., USA, 2015.

[3] Matlab, *Matlab User Manual*, Mathworks Inc.: Lowell, MA, USA, 2015.

[4] K.K. Nair, A.S. Kiremidjian, K.W. Law, *Time series-based damage detection and localization algorithm with application to the ASCE benchmark structure*, Journal of Sound and Vibration, 291, 349-368, 2006.

[5] A. Spagnoli, L. Montanari, B. Basu, B. Broderick, *Nonlinear damage identification in fiber-reinforced cracked composite beams through time-space wavelet analysis*, Procedia Material Science, 3, 1579-1584, 2014.

[6] S. Casciati, L. Elia, *The potential of Firefly Algorithm for damage localization and stiffness identification*, Studies in Computational Intelligence, 585, 163-178, 2015.

[7] C. Papadimitriou, *Optimal sensor placement methodology for parametric identification for structural systems*, Journal of Sound and Vibration, 278, 923-947, 2004.

[8] S. Casciati, *Stiffness identification and damage localization via differential evolution algorithms*, Structural Control & Health Monitoring, 15, 439-449, 2008.

[9] S. Casciati, L. Elia, *Potential of two metaheuristic optimization tools for damage localization in civil structures*, Journal of Aerospace Engineering, submitted, 2015.

[10] S. Casciati, L. Elia, *Potential of metaheuristic methods for damage localization and stiffness identification*, Proc., OPT-i, International Conference on Engineering and Applied Sciences Optimization, Kos, Greece, 2014.

[11] S. El-Borgi, S. Choura, C. Ventura, M. Baccouch, F. Cherif, *Modal identification and model updating of a reinforcement concrete bridge*, Smart Structures and Systems, 1, 83-101, 2005.

[12] M. Savoia, L. Vincenzi, *Differential evolution algorithm for dynamic structural indetification*, Proc., ICOSSAR '05, Rome, Italy, Millpress, Rotterdam, 2005.

[13] A. Carpinteri, M. De Freitas, A. Spagnoli, *Biaxial/Multiaxial Fatigue and Fracture*, Elsevier, 2003.

[14] P.G. Malerba, *Ponte "De Gasperi" – dai bozzetti all'opera finita*, Ed. MUP (In Italian), 2006.

[15] O.S. Bursi, E. Cazzador, A. Ussia, *Probabilisic analysis of a twin deck curved cable-stayed bridge subjected to multiple inputs and corrosion*, Engineering and Structures, 105(15), 87-98, 2015.

Chapter 4 Optimization of sensor deployment on the Ting Kau Bridge

In this chapter, the application of a bio-inspired algorithm, with a highly nonlinear objective function, in order to find an optimal sensors deployment across a large civil engineering structure for its modal identification is presented. The Ting Kau Bridge (TKB) [1], a cable-stayed bridges situated in Hong Kong, is chosen as a case study. The results show that the proposed method can identify eigenvalues and eigenvectors, and the number of sensors can be reduced, without a significant loss of accuracy.

4.1 An overview on modal parameter identification

Structural health monitoring (SHM) is an active area of research in civil engineering. Several system identification techniques have been developed over the past few decades, and their application is growing with the availability of instrumentation on civil infrastructures [2].

Typically, the goal is to estimate parameters of a mathematical model of the structure under study. Parameters identification through dynamic measurements is a discipline originally developed in mechanical and aerospace engineering [3], [4], and [5], but in the context of civil engineering, the structures (such as bridges and buildings) behave with their own features.

The parameters to be estimated by dynamic measurements are mainly of a modal nature, such as frequencies, damping ratios, and mode shapes. They will serve as a basis for the input to the finite element modal updating, in detecting and locating damage, as well as in assessing structural safety under special scenarios, as for instance large earthquakes and wind loads. There are principally three types of structural dynamic testing:

(1) forced vibration testing,
(2) free vibration testing, and
(3) ambient vibration testing.

The first relies on artificial items, such as drop weights, vibrodynes, and shakers, to excite the structure. For large infrastructures these devices are either unavailable or too expensive. In the second class, a free vibration condition is induced, by assessing adequate initial conditions. The main drawback using both these techniques is that the traffic along the infrastructure has to be stopped for a rather long period. The third class of methods does not require the interruption of the service, because it uses the disturbance, either wind or traffic, as an excitation [6].

Typically, the modal parameter identification is carried out both with input and output measurements of data through the frequency response functions (FRFs) in the frequency domain and impulse response functions (IRFs) in the time domain. In civil engineering, sensors that are usually installed at different locations, and record the dynamic response (i.e. the output) of each structure.

By the way, quantifying the input or the excitation level of a real structure in its service conditions is a difficult task. Indeed, the measurement of the input excitation forces acting on large structure is not easy. The need to identify modal parameters under real operational condition sometimes occurs, because the quality of real operating conditions measurements of complex structures may differ from those obtained from controlled laboratory environments.

A further benefit from the output-only data is the saving in equipment, since no tools are needed to excite the structure. The ambient vibration measurement is an output data-only dynamic testing where natural excitations are represented by winds and typhoons and it is an attractive topic in the civil engineering field.

The output-only modal identification methods can be classified into two main groups, namely, frequency domain methods and time domain methods [7]. The major frequency domain methods include the peak picking method (PP) [8], the frequency domain decomposition (FDD) [9], and the enhanced frequency domain decomposition (EFDD) [10], [11]. While, time domain methods gather the random decrement technique (RDT) [12], [13], [14] the natural excitation technique (NExT) [15], [16], the eigensystem realization (ERA) [17], [18], the data-driven and covariance-driven stochastic subspace identification (SSI) techniques [19], [20], [21], [22], [23], [24], the autoregressive moving average model (ARMA) technique [25], [26], and so forth.

Furthermore, the ambient vibration testing has been adopted to many large-scale bridges. Amongst others, one can mention the Golden Gate Bridge where experimental investigations were conducted to determine parameters of major interest in both wind and earthquake problems, and tests that involves simultaneous measurements are dealt [27];
the Faith Sultan Mehmet Suspension Bridge where ambient accelerations due to the dynamic excitation by wind and traffic were measured on the deck, towers, cables and hangers for validating the mathematical modeling used in seismic analyses of the bridge [28]; the Tsing Ma Suspension Bridge where the modal analysis is performed to determine natural frequencies and mode shapes of lateral, vertical, torsional, longitudinal, and coupled vibrations of the bridge [29]; the Vasco da Gama Cable-Stayed Bridge where the dynamic tests performed on the basis of non-conventional testing system, comprehending several independent accelerographs [30], the Kap Shui Mun Cable-Stayed Bridge, in Hong Kong where a finite-element model is established and the results are compared with the ones recorded *in situ* [31]; the Roebling Suspension Bridge where a comparison between the recorded data and a finite-element model is carried out, obtaining as outcome the possibility in the preservation of the suspension bridge [32]; the steel girder arch bridge where the field test is carried out by ambient vibration testing under traffic and wind-induced excitations [33]; the Hakucho Suspension Bridge where different output-only methods are applied and their accuracy is investigated and then compared [34]; the Humber Bridge where the operational modal analysis technology (OMA) technique is applied and the identification of the system is carried out [35]; the Tamar Suspension Bridge [36] where the structural health monitoring practice is performed; and the Vincent Thomas Suspension Bridge where the study focuses on seismic vulnerability of the retrofitted bridge. [37].
Using ambient vibration testing only response data are measured, while the current loading condition is unknown. Hence, a modal parameter identification process based on output-data only is needed.
The modal analysis, involving output-only measurements, necessitates the usage of particular modal identification techniques, to deal with small magnitudes of ambient vibration contaminated by noise without the knowledge of input forces. Since last decades, the technique of experimental modal parameter identification in the civil engineering field has developed very fast.
So, studying the mechanism behind the output-only modal identification, deficiency in modal identifiability, and criteria to evaluate robustness of the identified modes is a very important challenge [55].

Herein, one aims to study the mechanism behind the output-only modal identification, deficiency in modal identifiability, and criteria to evaluate robustness of the identified modes, but also to apply various methods of output-only modal identification.
The stochastic subspace identification-data driven method (SSI-data) as implemented in MACEC [38] is used to identify modal parameters. The results are then combined with a metaheuristic bio-inspired tool, namely the Firefly Algorithm [39], and an optimal reduced sensors deployment is pursued. The aim is to identify the same number of modal parameters (thus showing the robustness of the achieved results), but with a more sustainable economic effort. A strongly nonlinear objective function that takes into account not only the eigenvalues, but also the eigenvectors is introduced, and this is why the solution of the optimization problem is searched via a metaheuristic algorithm.

4.2 Governing relations

Hereinafter, governing relations are briefly described in order to introduce the policy of sensors reduction.

4.2.1 Stochastic subspace identification

Ambient excitation testing requires elaboration by a modal parameter identification method that is able to deal with ambient vibration measurement. Among them, the authors selected the stochastic subspace identification (SSI) method, well-implemented in the computational tool utilized: MACEC [38] working within the software environment MATLAB® [40].
The literature provides the main references on SSI [41], [42], [43], [44].
A structural dynamic model is described by a set of linear second-order differential equation with constant coefficient:

$$\mathbf{M}\ddot{u}(t)+\mathbf{C}\dot{u}(t)+\mathbf{K}u(t)=g(t) \qquad (44)$$

where **M**, **C**, and **K** denotes the time-invariant mass, the damping and the stiffness matrices of the structure, respectively. These are associated with the *n* generalized coordinates including the vector $u(t)$, whereas $g(t)$ is a time-dependent vector of input forces. If a state-space representation is considered, Equation (45) is rewritten as a first-order system of differential equations:

$$\dot{x}(t)=Ax(t)+Bu(t) \qquad (45)$$

where $x(t)=\left[u(t),\ \dot{u}(t)\right]^{\mathrm{T}}$ is the state vector, *A* is the state matrix and *B* is matrix of the excitation coefficient. They are defined as follows:

$$A=\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \quad B=\begin{bmatrix} 0 \\ M^{-1}B_2 \end{bmatrix} \quad g(t)=B_2u(t) \qquad (46)$$

Moreover, the output vector, $y(t)$, can be a part, or a linear combination of system states as:

$$y(t)=Cu(t)+Du(t) \qquad (47)$$

where *C* and *D* are the real output influence coefficient matrix and the out control influence coefficient matrix, respectively. Both Equations (45) and (47) provide a continuous-time state-space model of the dynamic system. The sample time and noise are always influencing the measurements. Hence, after the sampling such a model modifies into:

$$\begin{aligned} x_{k+1} &= \hat{A}x_k+\hat{B}u_k \\ y_k &= Cx_k+Du_k \end{aligned} \qquad (48)$$

Where $x_k=x(k\Delta t)$ is the discrete time state vector, $\hat{A}=\exp\left(A\Delta t\right)$ is the discrete state matrix, $\hat{B}=\left[\hat{A}-I\right]A^{-1}B$ is the discrete input matrix. Then, Equation (48) represents a discrete-time state-space model of a dynamic system.

A further issue is represented by the process noise due to the disturbance and the modeling inaccuracies. Thus, if the stochastic components, i.e. the noise, are included, Equation (48) can be extended to consider also a process noise w_k and a measurement noise v_k drawn as a continuous-time stochastic state-space model:

$$\begin{aligned} x_{k+1} &= \hat{A}x_k+\hat{B}u_k+w_k \\ y_k &= Cx_k+Du_k+v_k \end{aligned} \qquad (49)$$

Since it is difficult to find the individual process and measurement noise in an accurate way, some assumptions have to be made. Thus, the process noise w_k and the measurement noise v_k are assumed to be of zero-mean, white, and with covariance matrices as:

$$E\left[\begin{pmatrix} w_p \\ v_p \end{pmatrix}\left(w_q^{\mathrm{T}}v_q^{\mathrm{T}}\right)\right]=\begin{pmatrix} Q & S \\ S^{\mathrm{T}} & R \end{pmatrix}\delta_{pq} \qquad (50)$$

where *E* denotes the expected value operator and δ_{pq} is the Kronecker delta. Both sequences w_k and v_k are assumed statistically independent of each other.
Dealing with practical civil engineering issues, only the responses of a structure are measured, while the input sequence u_k remains unmeasured. When ambient vibration tests are performed, it is impossible to distingue the input term u_k from the noise term w_k and v_k. The result is a purely stochastic system, as expressed in:

$$\begin{aligned} x_{k+1} &= \hat{A}x_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \qquad (51)$$

The input is now implicitly modeled by the noise terms (the second terms in the right hand side of the above equation). Anyway, any assumption related to the white noise has to be explicit. The consequence reveals that such the assumption is violated.
Equation (51) becomes the basis for the time-domain system identification through ambient vibration measurements. The subspace method is able to identify the state space matrices based on the measurements using the QR-factorization, singular value decomposition (SVD), and least squares (LS), as numerical techniques. Thus, the QR-factorization results in a significant data reduction, while the SVD rejects the noise. Once the mathematical description of a structure, i.e. the state space model, is defined, the modal parameters are determined: natural frequencies, damping ratios, and mode shapes.

4.2.2 The MACEC software

Modal analysis of a structure develops along three principal steps that are the data collection, the system identification and the determination of modal features, such as eigenvalues, damping ratios, mode shapes and so forth. MACEC, a toolbox of MATLAB® [40] is a powerful tool developed by the Catholic University of Leuven in Belgium [38] that manages with every step in the modal analysis procedure, save for the data collection.
Such tool is herein applied in order to verify the modal parameter under the first set of blind-data provided for this benchmark study.

4.2.3 A bio-inspired approach for structural optimization

The bio-inspired swarm intelligence method, namely the Firefly Algorithm (FA), developed observing the social behavior of fireflies [45] was adopted and developed. As dealt in Chapter 1, such algorithm is gradient-free and it is very useful in solving optimization

issues with strong non-linearity. One of the greatest advantage of these tools, is that they do no trap in any local minimum or maximum, as occurring for the same basic Genetic Algorithms in [46] and [47], thus reaching the best value of the objective function. This novel method was first introduced for continuous optimization [39] and later extended to discrete problems such as structural control, where among the others one can find in [48], [49] and [50].
In general, FA combines three main strategies: attractiveness, brightness, and distance between each firefly and it can be idealized with three main assumptions. Basic rules of the Firefly Algorithm are described in Section 1.3.3.

4.3 A policy for sensors reduction

Modal analysis of structure develops along three principal steps, which are the data collection, the system identification, and the determination of modal features, as eigenvalues, damping ratios, mode shapes, and so forth. MACEC [38], a MATLAB® [40] toolbox, is a powerful tools enhanced by researchers at Catholic University of Leuven in Belgium. This tool manages with every step in the modal analysis procedure, save for the data collection.
Such toolbox is herein applied for verifying the modal parameter under the blind-data provided by the Hong Kong Polytechnic within a benchmark study.

4.3.1 Problem statement

In this Chapter, the actual sensor deployment on the bridge deck has been take into account and the possibility of reducing the number of sensors maintaining enough modal information is pursued. Indeed, one of the goals to be achieved consists on identifying the second mode when normal excitation are occurring. At this stage, this mode is detected only typhoon conditions.
As from Figure 4.1, where the system architecture of the proposed method is sketched, one has to specify:

a) the input of the optimization tool;
b) the objective function.

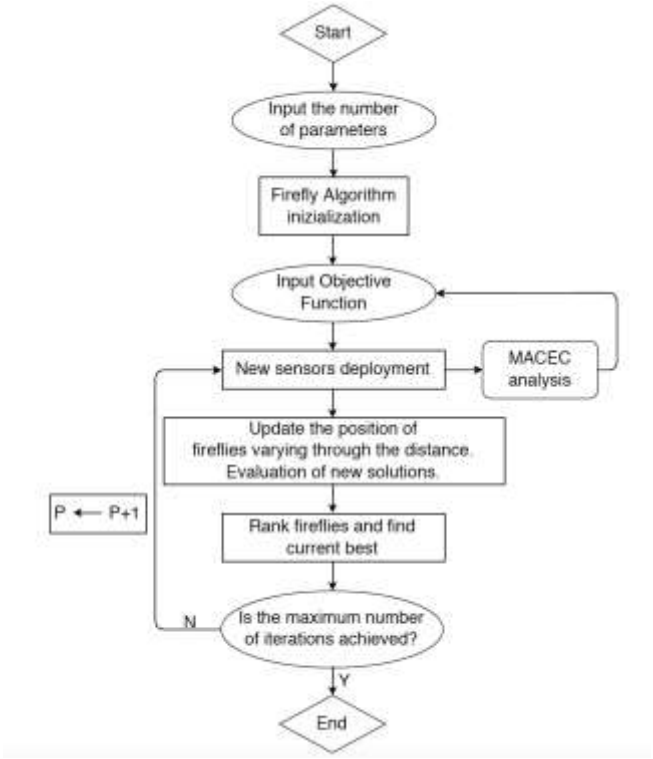


Figure 4.1 Proposed system architecture

4.3.2 Symbolism

Dealing with SHM, means evaluating the performance of a structure, appreciating its dynamic response after it has been excited. Several and well-known modal analysis tools, usually employed for linear systems, are applied for achieving the modal features, for instance the frequencies and the mode shapes.

For dealing with the inverse problem, a numerical approach, which consists of performing a finite element analysis where some variables have to satisfy the requirement of minimizing the discrepancies with the measured response, is proposed. For instance, an objective function, which is minimized when the difference of the measured and generated modal parameters, is formulated [51].

Traditional modal analysis can be applied to linear system to reach their modal features (i.e., natural frequencies and mode shapes). Such parameters are denoted with the subscript '*F*' because they refer to the scenario where the entire set of sensors is placed at the

bridge deck. Namely, $\bar{\boldsymbol{\omega}}_F$ denotes the $N\times 1$ vector of frequencies, whereas $\bar{\boldsymbol{\Phi}}_F$ the $N\times N$ of corresponding modal shapes.

By contrary, the scenario where only a lower number of sensors is considered is stated with the subscript '*NF*', where both eigenvalues and eigenvectors are stored in a $N\times 1$ vector ($\bar{\boldsymbol{\omega}}_{NF}(\boldsymbol{x})$), and in a $N\times N$ matrix ($\boldsymbol{\Phi}_{NF}(\boldsymbol{x})$), respectively.

4.3.3 The objective function

As widely described in Section 3.1, the objective function can be easily written in its scalar form by changing the subscript, and underlining the difference between the full and not full scenarios as:

$$F(\boldsymbol{x})=\sqrt{\sum_{i=1}^{N}\frac{1}{i}\left(\frac{\bar{\omega}_{i,F}-\bar{\omega}_{i,NF}(\boldsymbol{x})}{\bar{\omega}_{i,F}}\right)^2}+w\max_{1\le j\le N}\left[\frac{\sum_{i=1}^{N}\left(\bar{\Phi}_{ij,F}-\bar{\Phi}_{ij,NF}(\boldsymbol{x})\right)^2}{\sum_{i=1}^{N}\bar{\Phi}_{ij,F}^2}\right] \quad (52)$$

where the *i*-th element is given by $1/i$ which prioritizes the lower frequencies over the higher ones disturbed by measurement noise.

Also for this example, the second term of Equation (52) revealed very important, because, after a weighted calibration, the convergence of the method enhanced. This statement is confirmed by the numerical result presented in the following sections. Then, a weight function, *w*, is introduced and set by a 'trial and error' procedure, until satisfying results are achieved in term of convergence.

Finally, the optimization problem is posed as follow:

$$\begin{aligned}&\text{minimize } F(\boldsymbol{x})\\&\text{under the constraint: } \boldsymbol{x}_{Lb}\le\boldsymbol{x}\le\boldsymbol{x}_{Ub}\end{aligned} \quad (53)$$

where $\boldsymbol{x}_{Lb}$ and $\boldsymbol{x}_{Ub}$ are the $ns\times 1$ vectors of the lower and the upper bounds of each variable in the design parameter space, respectively.

Since the problem deals with high nonlinearity, the minimization of the objective function in Equation (52) is reached by the adoption of a metaheuristic tool, namely the Firefly Algorithm (FA).

4.4 The Ting Kau Bridge

4.4.1 Actual deployment of devices

In the last few decades, cable-stayed bridges (CSBs) and footbridges (CSFs) have experienced a tremendous spread, and increasingly challenging realizations are reported by practitioners and researchers worldwide [52], [53], and [54]. The Ting Kau Bridge (TKB) is a three-tower cable stayed bridge situated in Hong Kong which spans the Tsing Yi Island to the Tuen Mun Road [55]. The two central spans are 448m and 475m long, respectively, and there are two side-spans of length 127m,
Figure 4.2 shows an aerial view of the bridge, while Figure 4.3 illustrates the general layout of the bridge under study. The bridge deck is divided into two carriageways of 18.8m width. Along the deck there are three slender single-leg towers with 170m, 198m, and 158m respectively. Two steel girders along the edges of the deck with steel crossgirders every 4.5m, and a concrete slab on the top form each carriageway.

Figure 4.2 Aerial view of Ting Kau Bridge (TKB)

Furthermore, there is a 5.2m gap between the two parallel carriageways: they are linked each to the other every 13.5m by connecting crossgirders. Finally, 384 stay cables in four cable planes support the deck. The bridge has a unique feature, which consists in the arrangement of the three single-leg towers, strengthened by longitudinal and transverse cables, whose function is stabilizing.
A unique feature of the bridge consists in the arrangement of the three single-leg towers, strengthened by longitudinal and transverse cables, with a stabilizing function. There are 8 longitudinal stabilizing cables used to diagonally connect the top of the central tower to

the Ting Kau and Tsing Yi Towers (the length reaches 465m), whereas 64 cables are utilized to strengthen the three towers in the transverse (lateral) direction [55], [56].

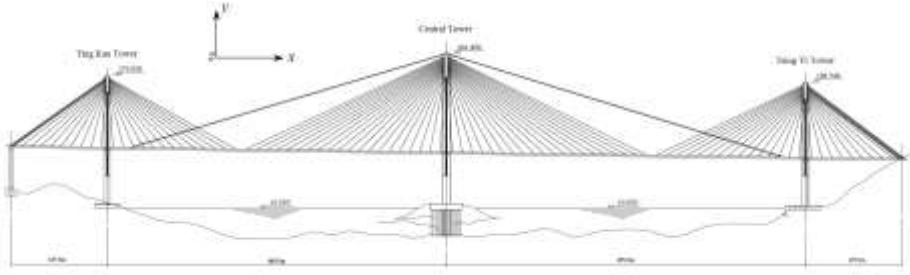


Figure 4.3 View of Ting Kau Bridge (TKB)

During the bridge construction and also after its completion in 1999 [57], [58], a number greater than 230 sensors has been installed on the TKB, within a long-term SHM system conceived by the Hong Kong SAR Government Highways Department. On the bridge are deployed several devices, as accelerometers, anemometers, strain gauges, temperature sensors, GPS, and weigh-in-motion sensors [59], [60]. 24 uniaxial, 20 biaxial, and 1 tri-axial accelerometers are permanently installed on the deck of the two main spans and two side spans, the longitudinal stabilizing cables, the top of three towers, and the base of the central tower. They form a total of 67 accelerometer channels and the monitor the dynamic response of the bridge itself. Herein, only the data collected by the devices (accelerometers) installed on the bridge deck are considered.

Figure 4.4 illustrates the placement of accelerometers and anemometers at the bridge deck, within the general layout of the bridge.

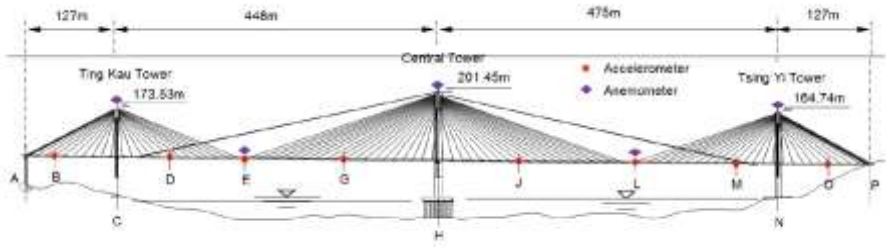


Figure 4.4 Deployment of accelerometers and anemometers at the bridge deck

In each of the sections from *A* to *P* in Figure 4.4, two accelerometers are installed on the east and west side of the longitudinal steel girders, respectively. They measure the vertical acceleration, while another accelerometer is installed on the central crossgirder and measures the transverse acceleration. The sampling frequency is 25.6Hz. Furthermore, 7

anemometers are installed at the top of each of the three towers and two on two main spans. Ultrasonic anemometers are installed on the east and west side of the deck and the sampling frequency of each anemometer is 2.56Hz.
Figure 4.5 shows the actual deployment of accelerometers at the bridge deck.

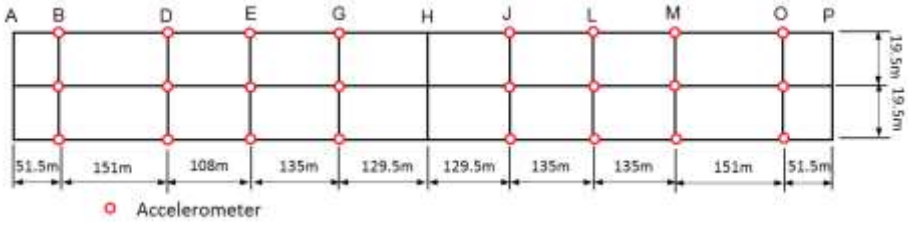


Figure 4.5 Actual deployment of accelerometers at the bridge deck

4.4.2 Existing recorded data

The benchmark on the data collected in 10 years of monitoring on the Ting Kau Bridge was launched in [55] and its companion document posted in the web [61]. Initially this latter document was listing the identified frequencies in Table 4.1, from where some frequencies were removed as spurious in the currently appended document. They are marked by a star in the table. The frequencies were identified having available 6 sets of data under weak wind conditions, 5 sets of data under typhoon conditions and other sets of monitoring acceleration data (also called "blind dataset") coming without any specification on the excitation conditions. These data were collected in different period and under different wind speeds duration. The first set of these blind data was utilized in this purpose order to explain the properties of output-only methods.

4.4.3 The identified modal frequencies

The data driven stochastic subspace identification (data-driven SSI) technique is considered the most powerful class of the known identification techniques for natural input modal analysis in the time domain [62]. Such technique has been applied to identify the modal frequencies and modal shapes of the Ting Kau Bridge and it works directly with the recorded time domain signals. It is able to identify the space models from the output data only by the application of robust numerical techniques among which the QR factorization, the singular value decomposition and least squares, once the formulation of the state space model is achieved used with the 24 accelerometers in Figure 4.4, the identified frequencies are listed in Table 4.1.

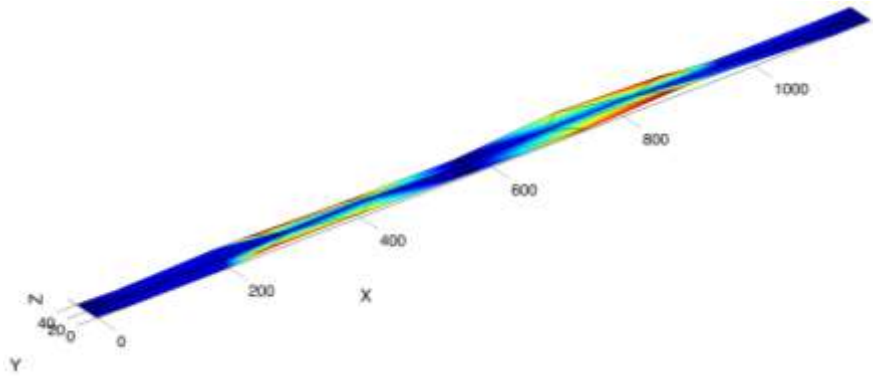
Next section provides more details on the technique in view of the further bits of information it makes available.

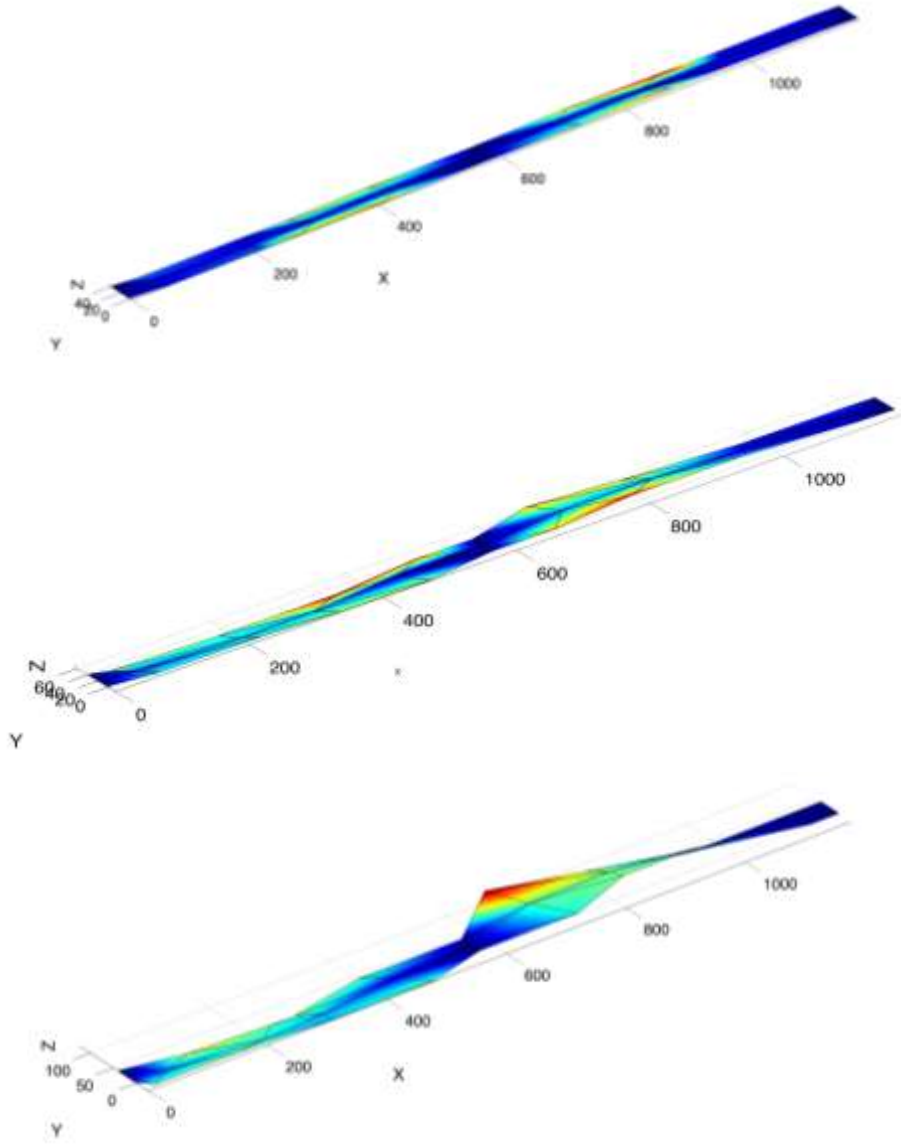
4.4.4 From the actual situation to the reduced one

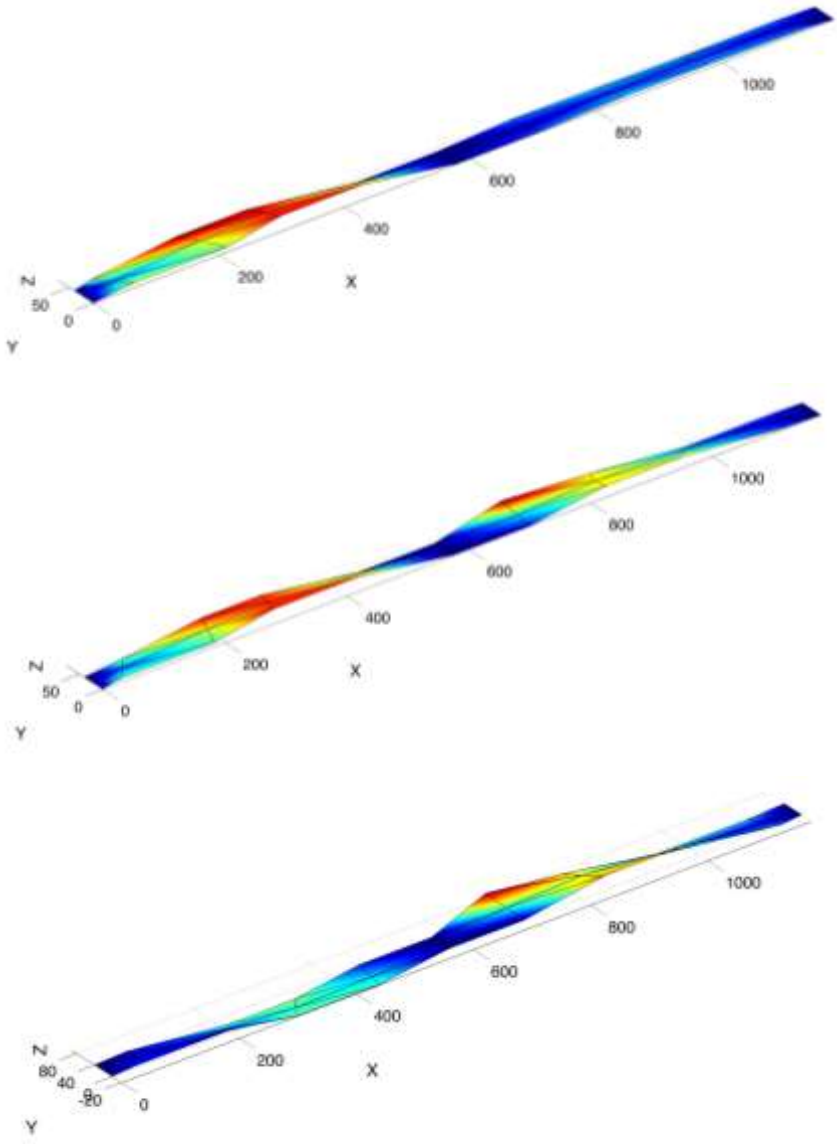
In Section 4.2, the actual deployment of the devices was provided. The current deployment permits one to carry out a general analysis, which by the software MACEC. The attention is focused on the first five eigenvalues as stated in Table 4.1.
In each analysis, both eigenvalues and eigenvectors are the output quantities parameters. Assume now that one relies on 16 sensors only. For the deployment in Figure 4.5, the eigenvalues in Table 4.1 are obtained by the same algorithm, together with the corresponding eigenvectors.

Table 4.1 First eight eigenvalues and damping ratios from the first blind-set data for 24 sensors (records of duration 1 hour without a non-particular condition of external excitation) extracted by MACEC and compared with Ni *et al.* 2015.

Eigenvalue	Frequency [Hz] (MACEC)	Frequency [Hz] (Ni *et al.*, 2015)	Damping ratio [%] (MACEC)
1	0.160*	-	1.34
2	0.162	0.162	2.85
3	0.178*	-	4.63
4	0.223	0.226	2.68
5	0.268	0.257	4.93
6	0.286	0.288	4.46
7	0.307	0.300	1.91
8	0.357	0.358	2.46




1000
800
600
400
200
X
0
Y
1000
800
600
400
200
0
Y
1000
800
600
400
200
X
100
50
0
0
Y


1000
800
600
400
200
X
Z
50
0
0
Y
1000
800
600
400
200
X
Z
50
0
0
Y
1000
800
600
400
200
X
Z
80
40
0
-20
0
Y

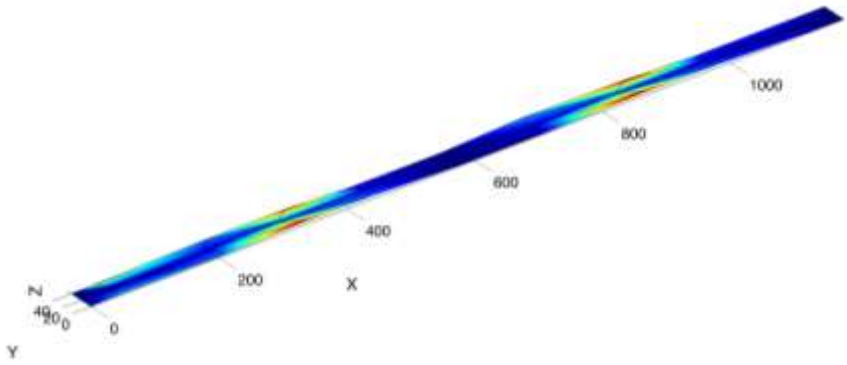


Figure 4.6 First 8 eigenvectors in case of 24 sensors at the bridge deck

4.5 Proposed method

4.5.1 Study on the actual situation

The control parameters of the adopted bio-inspired metaheuristic tool are summarized in Table 4.2. The randomization number, the minimum attractiveness and the absorption coefficient are maintained constant for each analysis performed, as suggested in the most of the implementations present in literature.

Table 4.2 Control parameters of Firefly Algorithm

Control parameter	Adopted value
NP, size of the initial population of fireflies	100
I_{max}, maximum number of iterations	40
ζ, randomization number	0.5
β_{min}, minimum attractiveness	0.2
γ, absorption coefficient	1.0

After the parameters have been set and under these assumptions, the solution of the problem is reached with a small computational burden.

4.5.2 Use of MACEC on the whole set of data

First, the grid, the number and the position of the degrees of freedom, and the type of finite element used for performing the analyses (beam or surface, depending on the selected type of finite element) are set. The implementation of the bridge deck in MACEC environment, including the features above described, is illustrated in Figure 4.7.

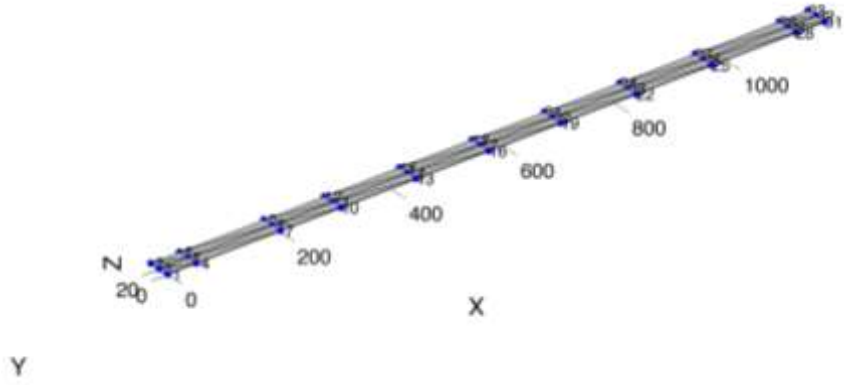


Figure 4.7 Bridge deck configuration in MACEC environment

After a phase where the signal is processed, the modal parameters are extracted from the modal analysis toward a stabilization diagram Figure 4.8 shows such diagram, which contains all the modal parameters. Figure 4.8 also shows a comparison between the full stabilization diagram and the stabilization diagram where only stable modes are denoted. In the upper figure, different dots represent both stable modes and modes that satisfy all stabilization criteria except for the damping and mode shape differences. While in the lower figure, modes that fulfill all the criteria are represented, and they coincide with the set provided within the benchmark study. Then the selected first eight eigenvalues are extracted, and compared with the ones found by exploiting the first blind-dataset provided by the Hong Kong Politechnic.

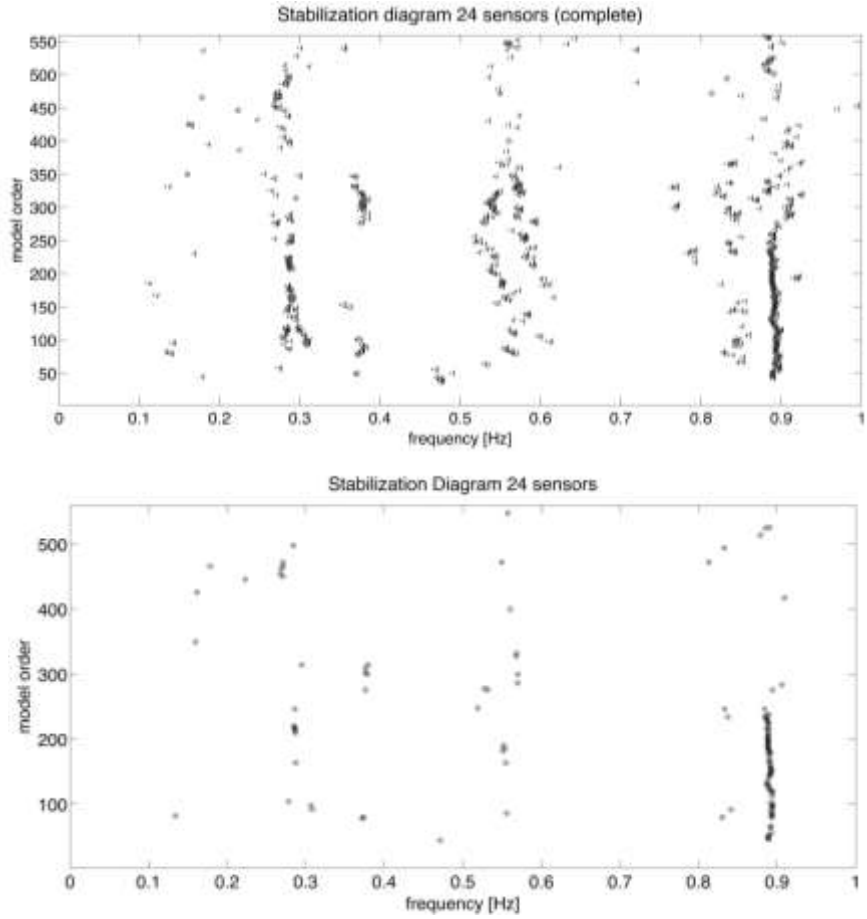


Figure 4.8 Stabilization diagrams for 24 sensors

Before applying this step, robust numerical techniques as the QR factorization and the singular value decomposition (SVD) are applied and the singular values for the maximum system order are obtained. For sake of completeness, each analysis is carried out on a Windows® 7 notebook, 64-bit, 2.67GHz Intel® Core™ i7 processor with 4GB ram.

4.5.3 Reducing the number of sensors on the bridge deck

Once the complete model is defined, i.e. with 24 sensors, the process of reduction that consists in the elimination of some sensors (for instance 8 sensors) starts. These sensors can considered superfluous, but the new deployment guarantees efficiency in terms of collection of modal parameters.

Furthermore, this purpose wants to minimize the economic effort, but always maintaining the lifeguard and the control of the structure.
The adopted procedure for the sensor deployment is the same to the previous one explained, and the algorithm parameters are kept constant for each analysis.
According to the objective function, the 16 sensors model is compared with the 24 sensors one and several deployment are proposed.
The best solution consider a value of the objective function and a deployment vector as shown in the following statement:

$$F\left(x_{sens}\right) = 0.0793$$
$$x_{sens} = [2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 16\ 17\ 18\ 19\ 21\ 22\ 24] \quad (54)$$

Such new configuration guarantees, albeit with a minimum error, to note the modal parameters and hence having a minimized economic impact. The new stabilization diagram and the new sensors deployment are shown in Figure 4.9 and Figure 4.10, respectively.

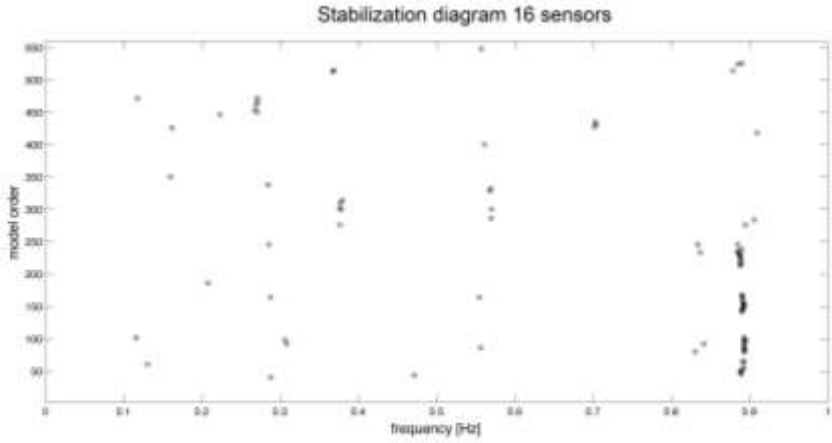


Figure 4.9 Stabilization diagram for 16 sensors

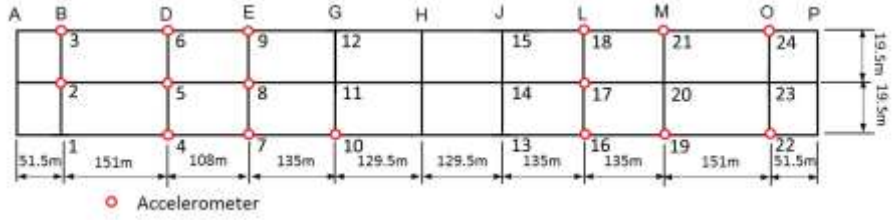


Figure 4.10 New proposed deck sensors configuration from MACEC

For such configuration, also the value of the damping ratios are shown in Table 4.3.

Table 4.3 Identified modal damping ratios under unknown excitation for a reduced set of 16 sensors

Frequency [Hz]	Damping ratio [%]
0.161	1.46
0.167	2.94
0.171	4.63
0.228	3.10
0.267	4.87
0.283	5.19
0.292	5.37
0.382	5.21

Both Table 4.1 and Table 4.3 frame the first eight frequencies obtained from the deployment of 24 and 16 sensors respectively, under the same excitation (i.e., the first set of the blind-data). Such these values can be comparable and one can observe that also reducing the number of sensors, the second mode is identified, so one of the challenge of this benchmark is correctly pursued.
For the evaluation of the eigen-properties of the system, namely frequencies and damping ratios as shown in Table 4.3, the approach presented in [22] has been adopted.
Finally, the path to convergence of the objective function is shown in Figure 4.11.

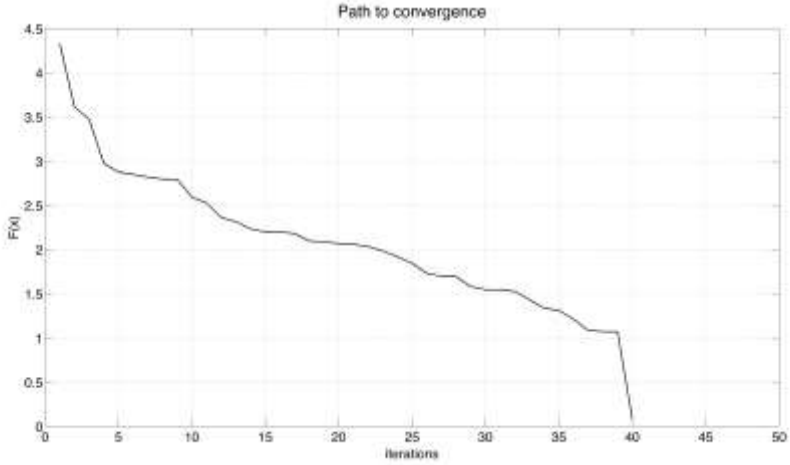


Figure 4.11 Path to convergence

Concluding, an application to a modal analysis method with a bio-inspired metaheuristic algorithm is implemented and applied with a highly nonlinear objective function in order to find an optimal sensor deployment across a large civil engineering structure. The results have shown that the proposed method identifies the frequencies, even adopting a reduced number of sensors. In terms of computational burden and convergence, the performance of the adopted optimization tool is fully satisfactory.

4.6 References

[1] Y.Q. Ni, Y.W. Wang, Y.X. Wang, *Investigation of the mode identifiability of a cable-stayed bridge: comparison from ambient vibration responses and from typhoon-induced dynamic responses*, Smart Structures and Systems, 15(2), 447-468, 2015.

[2] F. Vicario, M.Q. Phan, R. Betti, R.W. Longman, *Output-only observer/Kalman filter identification (O^3KID)*, Structural Control and Health Monitoring, 22(5), 847-872, 2015.

[3] D.J. Ewins, *Modal Testing: Theory and Practice*, Research Studies Press Ltd., England, 1986.

[4] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1987.

[5] J.N. Juang, *Applied System Identification*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

[6] W.X. Ren, T. Zhao, I.E. Harik, *Experimental and analytical modal analysis of a steel arch bridge*, J. of Struct. Engrg., 130(7), 1022-1031, 2004.

[7] B. Peeters, G. De Roeck, *Stochastic system identification for operational modal analysis: a review*, J. Dynam. Syst. Measurement Control, 123(4), 659-667, 2001.

[8] J.S. Bendat, A.G. Piersol, *Random Data: Analysis and Measurement Procedures*, John Wiley & Sons, New York, USA, 1986.

[9] R. Brincker, L. Zhang, P. Andersen, *Modal identification of output-only systems using frequency domain decomposition*, Smart Mater. Struct., 10(3), 441-445, 2001.

[10] S. Gade, N.B. Møller, H. Herlufsen, H. Konstantin-Hansen, *Frequency domain techniques for operational modal analysis*, Proc., 1st International Operational Modal Analysis Conference, edited by R. Brincker and N. Møller, Copenhagen, Denmark, 2005.

[11] N.J. Jacobsen, P. Andersen, R. Brincker, *Using enhanced frequency domain decomposition as a robust technique to harmonic excitation in operational modal analysis*, Proc., ISMA Conference on Advanced Acoustics and Vibration Engineering, Leuven, Belgium, 2006.

[12] S.R. Ibrahim, *Random decrement technique for modal identification of structures*, J. Spacecraft Rockets, 14(11), 696-700, 1977.

[13] J.K. Vandiver, A.B. Dunwoody, R.B. Campbell, M.F. Cook, *A mathematical basis for the random decrement vibration signature analysis technique*, J. Mech. Design, 104(2), 307-313, 1982.

[14] J.C. Amussen, *Modal analysis based on the random decrement technique*, PhD Thesis, Department of Building Technology and Structural Engineering, Aalborg University, Aalborg, Denmark, 1997.

[15] G.H. James III, T.G. Carne, and J.P. Lauffer, *The natural excitation technique (NExT) for modal parameter extraction from operating wind turbines*, Int. J. Anal. Exper. Modal Anal., 10(4), 260-277, 1995.

[16] P. Barney, T. Carne, *Modal parameter extraction using natural excitation response data*, Proc., 17th International Modal Analysis Conference, Orlando, Florida, USA, 1999.

[17] J.N. Juang, R.S. Pappa, *An eigensystem realization algorithm for modal parameter identification and model reduction*, J. Guid. Control Dynam., 8(5), 620-627, 1985.

[18] L.D. Peterson, *Efficient computation of the eigensystem realization algorithm*, J. Guid. Control. Dynam., 18(3), 395-403,1995.

[19] P. Van Overschee, B. De Moor, *Subspace algorithms for the stochastic identification problem*, Automatica, 29(3), 649-660, 1993.

[20] B. Peeters, G. De Roeck, *Reference-based stochastic subspace identification for output-only modal analysis*, *Mech. Syst. Signal Pr.*, **13**(6), 855-878, 1999.

[21] B. Peeters, G. De Roeck, *Stochastic system identification for operational modal analysis: a review*, J. Dynam. Syst. Measurement Control, 123(4), 659-667, 2001.

[22] E. Reynders, G. De Roeck, *Reference-based combined deterministic-stochastic subspace identification for experimental and operational modal analysis*, Mech. Syst. and Sign. Proc., 22(3), 617-637, 2008.

[23] C.H. Loh, Y.C. Liu, Y.Q. Ni, *SSA-based stochastic subspace identification of structures from output-only vibration measurements*, Smart Struct. Syst., 10(4-5), 331-351, 2012.

[24] Y.C. Liu, C.H. Loh, Y.Q. Ni, *Stochastic subspace identification for output-only modal analysis: application to super high-rise tower under abnormal loading condition*, Earthq. Eng. Struct. D., 42(4), 477-498, 2013.

[25] Y.L. Pi, N.C. Mickleborough, *Modal identification of vibrating structures using ARMA model*, J. of Engrg. Mech. – ASCE, 115(10), 2232-2250, 1989.

[26] P. Andersen, *Identification of civil engineering structures using vector ARMA models*, PhD Thesis, Department of Building Technology and Structural Engineering, Aalborg University, Aalborg, Denmark, 1997.

[27] A.M. Abdel-Ghaffer, R.H. Scanlan, *Ambient vibration studies of the Golden Gate Bridge. I: Suspended structure*, J. of Engrg. Mech. – ASCE, 111(4), 463-482, 1985.

[28] J.M.W. Brownjohn, A.A. Dumanoglu, R.T. Severn, *Ambient vibration survey of the Fatih Sultan Mehmet (Second Bosporus) suspension bridge*, Earthq. Engrg. Struct. Dyn., 21, 907-924, 1992.

[29] Y.L. Xu, J.M. Ko, W.S. Zhang, *Vibration studies of Tsing Ma Suspension Bridge*, J. Bridge Engrg. – ASCE, 2, 149-156, 1997.

[30] A. Cunha, E. Caetano, R. Delgado, *Dynamic tests on large cable-stayed bridge*, J. Bridge Engrg. – ASCE, 6(1), 54-62, 2001.

[31] C.C. Chang, T.Y.P. Chang, Q.W. Zhang, *Ambient vibration of long-span cable-stayed bridge*, J. Bridge Engrg. – ASCE, 6(1), 46-53, 2001.

[32] W.-X. Ren, I.E. Harik, M. Lenett, T. Basehearh, *Modal properties of the Roebling Suspension Bridge – FEM modeling and ambient testing*, Proc., IMAC-XX: A Conf. on Structural Dynamic, Kissimmee, Florida, February 5-8, 1139-1145, 2001.

[33] W.-X. Ren, T. Zhao, I.E. Arik, *Experimental and Analytical Modal Analysis of Steel Arch Bridge*, J. of Struct. Engrg., 130(7), 1022-1031, 2004.

[34] D.M. Siringoringo, Y. Fujino, *System identification of suspension bridge from ambient vibration response*, Engrg. and Struct., 30(2), 462-477, 2008.

[35] J.M.W. Brownjohn, F. Magalhaes, E. Caetano, A. Cunha, *Ambient vibration re-testing and operational modal analysis of the Humber Bridge*, Engrg. and Struct.,32(8), 2003-2018, 2008.

[36] K.Y. Koo, J.M.W. Brownjohn, D.I. List, R. Cole, *Structural health monitoring of the Tamar suspension bridge*, Structural Control and Health Monitoring, 20(4), 609-625, 2013.

[37] D. Karmakar, S. Ray-Chaudhuri, M. Shinozuka, *Finite element model development, validation and probabilistic seismic performance evaluation of Vincent Thomas suspension bridge*, Struct. and Infrastruct. Engrg., 11(2), 223-237, 2015.

[38] E. Reynders, G. De Roeck, M. Schevenels, *MACEC 3.3 User's Manual*, Department of Civil Engineering, Catholic University of Leuven, Belgium, 2015.

[39] X.-S. Yang, *Multiobjective Firefly Algorithm for Continuous Optimization*, Engineering with Computers, 29(2), 175-184, 2013.

[40] Matlab, *Matlab User Manual*, Mathworks Inc.: Lowell, MA, USA, 2015.

[41] P. Van Overschee, B. De Moor, *Subspace identification for linear systems: theory, implementation and applications*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[42] B. Peeters, G. De Roeck, *Reference-based stochastic subspace identification for output-only modal analysis*, Mech. Syst. Signal Pr., 13(6), 855-878, 1999.

[43] H.P. Chen, T.L. Huang, *Updating finite element model using dynamic perturbation method and regularization algorithm*, Smart Struct. and Syst., 10(4-5), 427-442, 2012.

[44] E. Reynders, G. De Roeck, *Reference-based combined deterministic-stochastic subspace identification for experimental and operational modal analysis*, Mech. Syst. and Sign. Proc., 22(3), 617-637, 2008.

[45] X.-S. Yang, *Nature-inspired metaheuristic algorithms, 2nd edition*, Luniver Press, 2010.

[46] S. Casciati, *Stiffness identification and damage localization via differential evolution algorithms*, Structural Control Health Monitoring, 15(3), 436-449, 2008.

[47] S. Casciati, *Differential evolution approach to realibility-oriented optimal design*, Probabilistic Engrg. Mech., 36, 72-80, 2014.

[48] S. Casciati, L. Elia, *Potential of two metaheuristic optimization tools for damage localization in civil structures*, J. of Aerospace Engrg., submitted, 2015.

[49] S. Talatahari, A.H. Gandomi, G.J. Gun, *Optimum design of tower structures using Firefly Algorithm*, Structural Design of Tall and Special Buildings, 23(5), 350-361, 2014.

[50] G.-D. Zhou, T.-H. Yi, H. Zhang, H.-N. Li, *Energy-aware wireless sensor placement in structural health monitoring using hybrid discrete firefly algorithm*, Structural Control Health Monitoring, 22(4), 648-666, 2015.

[51] S. Casciati, L. Elia, *Innovative, bio-inspired, metaheuristic methods to SHM diagnostic goals*, Proc. 7SHMII – Seventh International Conference on Structural Health Monitoring of Intelligent Structures, Turin, Italy, 2015.

[52] J. Strasky, *Stress ribbon and cable-supported pedestrian bridges*, ICE Publishing, Thomas Telford, London, 2011.

[53] H. Swensson, *Cable-stayed Bridges*, Ernst & Sohn Gmbh and Co., Berlin, Germany, 2012.

[54] O.S. Bursi, A. Kumar, G. Abbiati, Identification, *Model Updating, and Validation of a Steel Twin Deck Curved Cable-Stayed Footbridge*, Computer-Aided Civil and Infrastructure Engineering 29, 703-722, 2014.

[55] R. Bergermann, M. Schlaich, *Ting Kau Bridge, Hong Kong*, Struct. Eng. Int., 6(3), 152-154, 1996.

[56] Y.Q. Ni, Y.W. Wang, Y.X. Xia, *Investigation of mode identifiability of a cable-stayed bridge: comparison from ambient vibration responses and from typhoon-induced dynamic responses*, Smart Structures and Systems, 15(2), 447-468, 2015.

[57] K.Y. Wong, *Instrumentation and health monitoring of cable-supported bridges*, Structural Control & Health Monitoring, 11(2), 91-124, 2004.

[58] J.M. Koo, Y.Q. Ni, *Technology developments in structural health monitoring of large-scale bridges*, Eng. Struct., 27(12), 1715-1725, 2005.

[59] K.Y. Wong, *Design of a structural health monitoring system for long-span bridges*, Struct. Infrastruct. E., 2007.

[60] Y.Q. Ni, K.Y. Wong, Y. Xia, *Health checks through landmark bridges to sky-high structures*, Adv. Struct. Eng., 14(1), 103-119, 2011.

[61] Dept. of Civil and Environmental Engineering, The Hong Kong Polytechnic University, http://zn903.com/benchmarkstudy/, 2015.

[62] R. Brincker, P. Andersen, *Understanding stochastic subspace identification*, Proc., 24th International Modal Analysis Conference, St. Louis, Missouri, USA, 2006.

Conclusions

In this thesis, some mathematical tools, namely the metaheuristic algorithms, are described, implemented, and finally applied on case studies taken as examples.
The essential motivation behind the choice of these typology of optimization tool is that they are new methods and they can achieve satisfactory results in term of performance, which include the computational burden and the time to converge to the best solution.
This aspect becomes important because it is able to find and give new possibilities and also can lower the economic effort.
After a state-of-the-art of the heuristic and metaheuristic methods shaped on most used methods in the literature in Chapter 1, four methods are selected as solving tools, and tested with some benchmark mathematical functions in Chapter 2.
Consequently, in Chapter 3, the attention is point out on such metaheuristic algorithms applied on three case studies, such as a cantilever beam, a frame structure, and a bridge. These tools are exploited for detecting and localizing the damage, and so it becomes easier assessing some possibilities for the structural health monitoring and the control.
The fourth Chapter is devoted to the description of a large cable-stayed bridge, where one of the selected tool is applied for reducing the number of sensors but maintaining the structure in condition of security.

Acknowledgements

Mi trovo a scrivere la parte (forse) più difficile di questa tesi. Significa essere arrivati ad un traguardo e significa volere ringraziare tutte quelle persone che, negli ultimi tre anni, hanno speso del tempo con me e per me.
In prima battuta i Proff. Faravelli e Casciati che si sono sempre dimostrati disponibili e mi hanno aiutato a tracciare questo percorso triennale, e a maturare dal punto di vista accademico e personale.
Un grazie va sicuramente detto a Paolo e Marco, colleghi, ma soprattutto amici con i quali ho condiviso e continuo a condividere passioni e pensieri.
Grazie va ai miei colleghi d'ufficio Daniele, ma soprattutto Michele, con il quale c'è una amicizia sincera e duratura sin dai tempi dei primi esami universitari.
Grazie ai miei amici di sempre Ale ed Ale, con i quali basta una parola per capirsi e che qualsiasi cosa succeda ci sono.
Grazie a Chiara che è arrivata nella mia vita da appena un anno, ma è come se fosse con me da una vita. Dal primo momento è riuscita a capirmi, a starmi vicino, a spronarmi quando le cose non hanno preso il verso giusto, e a non farmi mai dimenticare di pormi degli obiettivi sempre più alti e a crescere sempre di più. Credo che poche volte nella vita possano capitare delle fortune così.
L'ultimo – solo in ordine di elenco – è il ringraziamento a Mamma, Papà e Andrea. Anche questa volta mi commuove scrivere queste ultime righe, ma davvero grazie. Perché è soprattutto grazie a voi tre che sono quello che sono (chissà se avete fatto davvero un buon lavoro!?).

Grazie!