Politecnico di Torino


**Master's Degree in CIVIL ENGINEERING**

# Structural Optimization with Swarm-based Algorithms

## Master Degree Thesis

**Supervisor:**
**Prof. G. C. Marano**

**Co-supervisor:**
**Ph.D. Student R. Cucuzza**

**Candidate:**
**Marco Martino Rosso**

**Student's ID:**
**S264386**

**October 2020**
**A.Y. 2019/2020**

**Abstract**

The purpose of the present dissertation is not only to introduce different methods for solving optimization problems, but also to expose a new strategy to deal with constrained structural optimization problems which combines modern Artificial Intelligence and machine learning methods. In the first introductory part, the classical resolution methods are exposed mainly focusing on the Lagrange multiplier rule and the most used mathematical programming methods of the past. These approaches, based on the calculation of the gradient of the objective function, usually solve optimization problems in a sequential way. Afterwards, an overview of the most popular modern meta-heuristic approaches to solve optimization problems is presented. Inspired by Nature, these methods are included within the framework of Soft Computing, which is a still very active research sub-field of Artificial Intelligence. The attention is mainly focused on the Particle Swarm Optimization (PSO) algorithm which is based on the emerging intelligent convergent behaviour of a swarm influenced by social and cognitive interactions that are inspired by natural bird flocking behaviour. Originally formalized in 1995, the PSO is nowadays still under study to improve the search process performances. For this reason, many variants and several constraint handling strategies have been developed during the recent years. Although the penalty functions still remain the most adopted indirect method to deal with constrained problems, due to their several drawbacks, in this Thesis a new non-penalty constraint handling approach is discussed. The PSO is thus combined with a constraint handling technique to preserve the feasibility of candidate solutions, which is based on the predictive model generated by the Support Vector Machine (SVM), a machine learning method usually adopted for classification problems. Because of its generality, the constraint handling with SVM appears to be more adaptive both to non-linear and discontinuous boundary. However, to improve the performance of the algorithm, e.g. when the feasible region is little and narrow, a relaxation function of the constraints is also adopted to enlarge the actual search region.

In the final part of the present work, to assess the convergence properties of the new PSO-SVM algorithm, two numerical benchmark literature problems, which statements are located into the Appendix A, are discussed and compared in terms of objective function value with the well-known meta-heuristic genetic algorithm (GA) and with a PSO penalty-based. Subsequently, two structural examples are discussed concerning respectively the size optimization of constant cross section simply supported beam and the shape and size optimization of a planar steel warren truss beam with hollow core section profiles. In particular, this latter example highlights not only the importance of the optimization process to get the highest structural performances with the minimum costs, but also that the optimization algorithms represent, from the technical point of view, an essential tool for civil engineers, which strongly influences the decision process during the design phase. Therefore, the optimization algorithms are valuable and essential tools, which support the designers to identify the best technical solution among the infinite possibilities.

**Sommario**

Il presente elaborato tratta alcuni dei più noti metodi di risoluzione dei problemi di ottimizzazione, presentando anche una nuova tecnica per risolvere problemi di ottimizzazione strutturale vincolata che adotta e combina i moderni metodi di Intelligenza Artificiale e le tecniche di machine learning. Nella parte iniziale introduttiva, vengono presentati i classici metodi di risoluzione come il metodo dei moltiplicatori di Lagrange e i metodi di programmazione matematica, i più adottati in passato. Tali approcci sono di solito basati sul calcolo del gradiente della funzione obiettivo e adottano spesso tecniche di risoluzione sequenziale. Successivamente, viene esposta una breve presentazione di alcuni dei più famosi moderni metodi euristici per la risoluzione di problemi di ottimizzazione. Ispirati dalla Natura, questi metodi fanno parte delle cosiddette tecniche di Soft Computing, un ramo di ricerca ancora molto attivo nell'ambito dell'Intelligenza Artificiale. La trattazione si concentra poi principalmente sull'algoritmo di ottimizzazione con sciami di particelle (PSO) che è basato sull'emergente comportamento intelligente di sciami di individui che convergono verso il punto di ottimo grazie a interazioni sociali e cognitive, le quali sono state direttamente mutuate dallo studio del comportamento di veri stormi di uccelli in natura. Formalizzato per la prima volta nel 1995, ancora oggi il PSO è oggetto di ricerca e sviluppo, principalmente per migliorare le prestazioni di ricerca dello sciame, a tale scopo molte varianti e diverse tecniche per la gestione del vincolo sono state sviluppate nel corso degli anni. Sebbene la gestione del vincolo basata sulle funzioni di penalità sia ancora oggi il metodo più largamente diffuso, esso presenta anche numerosi inconvenienti e, pertanto, in questa Tesi viene presentato un innovativo approccio per la gestione del vincolo. Il PSO viene quindi combinato con una nuova tecnica di gestione del vincolo per preservare l'ammissibilità delle soluzioni candidate basandosi su un modello predittivo generato da una macchina a vettori di supporto (SVM), una tecnica di machine learning usualmente adottata nei problemi di classificazione. Grazie alla sua generalità, la gestione del vincolo basata sul SVM sembra essere più adattiva anche per vincoli con contorni non lineari e/o discontinui. Tuttavia, per migliorare le prestazioni dell'algoritmo, ad esempio con regioni ammissibili di piccole dimensioni e/o molto strette, è stata adottata una funzione di rilassamento dei vincoli che consente di allargare l'effettivo spazio di ricerca anche a una piccola zona non ammissibile, ma considerando solamente quelle ammissibili come soluzioni possibili del problema.
Nella parte finale del presente elaborato, per verificare le proprietà di convergenza del nuovo algoritmo PSO-SVM, due problemi numerici di riferimento tratti dalla letteratura tecnica, le cui formulazioni sono riportate in Appendice A, sono stati studiati e confrontati in termini di funzione obiettivo con il ben noto algoritmo genetico (GA) e con il codice PSO dotato di funzioni di penalità. Successivamente, sono stati analizzati due problemi strutturali che riguardano rispettivamente l'ottimizzazione dimensionale di una trave a sezione costante in semplice appoggio e l'ottimizzazione di forma e dimensionale di una trave reticolare piana di tipo Warren con profili scatolati in acciaio. Nella fattispecie, quest'ultimo esempio ha messo

in luce non solo l'importanza che ricopre il processo di ottimizzazione nell'ottenimento del massimo rapporto di sfruttamento strutturale dell'acciaio con il minimo consumo di materiale, ma, soprattutto, il fatto che gli algoritmi di ottimizzazione, dal punto di vista tecnico-pratico, sono uno strumento essenziale per gli ingegneri civili in quanto possono influenzare enormemente il processo decisionale durante le fasi di progettazione e, pertanto, svolgere una essenziale funzione di supporto per i progettisti al fine di individuare, tra le infinite scelte, la miglior soluzione tecnica possibile.

*To the people I love*
*Alle persone a cui tengo*

# Acknowledgements

First and foremost, I would like to express my sincere gratitude to professor G. C. Marano for giving me this opportunity to work with him, for guiding me and firstly introducing me to the fascinating world of the Structural Optimization field with particular interest to the new modern technologies. I should also express my thanks to the PhD student Eng. R. Cucuzza for his valuable help and his careful revision work.

Furthermore, I would like to dedicate the present Dissertation to all the people who love me. In particular, I would like to thank my family for always supporting me during the University years allowing me to cross the finish line in the best possible way. I am grateful to my mum Samai and my dad Giuseppe for being wonderful parents and for their immeasurable moral and economic support. I would like to thank also my brothers and sister, Roberta, Davide and Danilo, and their respective families for their support. I would like to warmly thank especially my nephews Leonardo and Gabriele because they have filled my heart with their smiles for over a year.

I wish a natural and honest gratitude to my girlfriend Simona, who has always been there for me during all these years. Substantially, we grow up together during the adolescence, enriching each other during the years, facing together all the obstacles and sharing all the joyful moments of our youth. I would like also to thank her for the revision work and for helping me to improve my English exposition. A special consideration to her dad Bruno, her mum Laura, her sister Marzia with the husband Enrico for always supporting me and, in particular, to my nieces Melissa and Noemi for their love towards me.

Last but not least, I express sincere gratitude to all my University mates with whom I shared a lot and who have always been for me a point of reference both from a professional and human perspective.

# Ringraziamenti

Innanzitutto, è doveroso ringraziare vivamente il professor G. C. Marano per avermi offerto questa opportunità di collaborazione, per avermi trasmesso la sua passione e la sua energia, per avermi seguito e guidato introducendomi al mondo dell'ottimizzazione strutturale e alle più moderne potenzialità tecnologiche oggi disponibili in ambito ingegneristico. Ringrazio intensamente anche il dottorando Ing. R. Cucuzza per il suo preziosissimo aiuto, per i numerosi momenti di confronto e per il suo curato lavoro di revisione.
Vorrei inoltre dedicare il presente elaborato a tutte le persone che mi vogliono bene. In particolare, vorrei ringraziare la mia famiglia per avermi sempre sostenuto in questi anni di università consentendomi di tagliare questo importantissimo traguardo nel migliore dei modi. Un grazie speciale alla mia mamma Samai e al mio papà Giuseppe per i meravigliosi genitori che sono e per il loro incommensurabile aiuto sia morale, ma anche di sostegno economico in questi anni di studio. Ringrazio anche i miei fratelli, Roberta, Davide e Danilo, e le rispettive famiglie per il loro continuo supporto. Soprattutto ringrazio calorosamente i miei due nipotini, Leonardo e Gabriele, che da oltre un anno ormai riempiono di gioia le mie giornate con i loro sorrisi.
Un naturale e sincero omaggio va alla mia ragazza, Simona, che mi è sempre stata accanto in tutti questi anni fin dalla scuola superiore. Siamo praticamente cresciuti insieme arricchendoci vicendevolmente per tutti questi anni, affrontando insieme e con amore anche i momenti di difficoltà e vivendo appieno i momenti più felici di questa nostra gioventù. Altresì, la ringrazio intensamente per il suo prezioso contributo nella revisione di alcune parti di questo lavoro e, nella fattispecie, per i suoi preziosissimi consigli che mi hanno aiutato a migliorare la qualità dell'esposizione in lingua Inglese. Un ringraziamento speciale va anche a suo papà Bruno, sua mamma Laura, sua sorella Marzia con il marito Enrico per il loro costante e sentito appoggio ma soprattutto alle mie nipotine Melissa e Noemi per il loro amore per me.
In ultimo, ma non per importanza, vorrei altresì ringraziare tutti i miei compagni di università con cui ho condiviso davvero tanto e che sono sempre stati per me un punto di riferimento sia professionale sia dal punto di vista umano.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The structural engineering problems can be roughly subdivided into modelling, simulation and optimization problems, as stated in [22]. The modelling problems are related to the formulation of a reliable description of the mechanical behaviour of the structure under certain load conditions and compatibility constraints through analytical or numerical formulations. This involves not only the accurate identification of the stress-strain laws (constitutive laws) of the materials and the failure criteria, but also the inverse problem, e.g. the dynamic identification of existing structures. Simulation problems involve the performing of structural safety analysis through the reproduction of the structural response under known load conditions in order to assess the performance of the structures, e.g. in the seismic analysis. The aim of the optimization problems is to find the best solution for a certain problem. When the problem is the cost optimization, the aim is to reduce the amount of material usage in order to lessen the related costs, whereas, in the performance optimization problems, the purpose is to find the best way to situate the material in order to obtain the highest possible rate in the structural performances. As stated in [22], the optimization problems can be further divided into control problems, related to the definition of the external actions which are able to lead to a specific desired response, or synthesis problems is the design optimization.

An optimization problem is nothing more than the solving of a minimization problem of a function which is called objective function (OF) or merit function $f(\boldsymbol{x})$ which depends on a certain choice of design variables or decision variables $\boldsymbol{x}$ . When the OF is constrained, the problems are denoted as constrained optimization problems, otherwise they are called unconstrained optimization problems. Without loss of generality, a maximization problem can be converted into a minimization one referring to the opposite of the OF $-f(\boldsymbol{x})$. An important property of the OF is that its behaviour is not affected if a positive constant is multiplied, divided or algebraically summed to it [64], therefore, in these cases, the optimum does not change its position. The structural optimization target is to find the best design, under certain points of view, with respect to other possible decisions. Based on the number of the goals an optimization problem wants to fulfil, the problems can be subdivided in: Single Objective optimization problems, in which only one OF is considered, and Multi-Objective problems, in which the optimal solution is a compromise among different OFs.

According to [9, 22], it is possible to define three main different fields in structural optimization problems: size optimization, shape optimization and topology optimization.
The target of *size optimization* is to find the optimal cross section of the structural elements subjected to stress and displacement constraints. Size optimization is also related to the cost minimization, because it aims to define the minimum usage of material. The design variables are the cross section dimensions of each structural member and they are usually discrete because limited by the use of commercial steel profile or related to construction limitations in site for RC structures [59]. Due to the complexity to deal with discrete problems, as stated in [13], the design variables are usually treated as continuous and, at the end of the optimization process, they are finally rounded to the nearest discrete possible value. In order to reduce the complexity of the problem, one possible strategy is to reduce the number of design variables e.g. gathering together the same typology of elements and associating the cross sections properties not to each structural member but to each group of elements [13]. An example of size optimization is presented in Figure 1.1 (a) in which the various elements of a cantilever truss beam have different sizes because they are stressed in different ways according to the load path toward the external restraints.

The *shape* or *geometry optimization* purpose is to find an optimal shape through the definition of some key points whose movements condition the overall shape of the structure with a fixed topology and/or some fixed boundary conditions [59, 13]. For example, considering a truss structure, the key points are the nodes where the members converge. Moving the node, all the connected elements have to adjust their length and inclination, therefore it condition the overall shape of the structure. The shape can change even maintaining e.g. the same topology (a specific type of truss structure), e.g. changing the height of a planar warren truss simply supported beam as depicted in Figure 1.1 (b). For some structures, considering the shape defined by the adoption of certain shape functions, it can be convenient to reduce the complexity of the problem or numerical efficiency, as shown in 1.1 (c). As reported in [13], in the past the form-finding problem was carried out with physical models. According to the Hook's principle, the model was realized by tied ropes with hanging weights (hanging models or inverted models) and, reversing the obtained funicular form, it was stated that the same shape could work as an arch in pure compressive state with the same loads reversed in sign. The most famous Spanish architect Gaudí used to study the form of its creations with this approach without any mathematical solving of the shape optimization problem [13]. Later on, the form-finding was studied with graphic static and nowadays with computer simulations and specific approaches [13].

The *topology optimization* not only aims to find the best spatial arrangement of the structural elements inside a certain fixed domain which influences the structural layout, but also analyses how the available material can be organized to obtain the best structural performances [59, 22]. As stated in [13], the topology optimization started in 1904, when the Australian researcher Michell studied the optimal layout of truss structures based on considerations about the isostatic lines and the strain field. For instance, for a fixed span, the topology optimization goal is to define the type of truss to use for the specific loads and boundary conditions. Another possible approach to perform topology optimization in continuum design space is related to

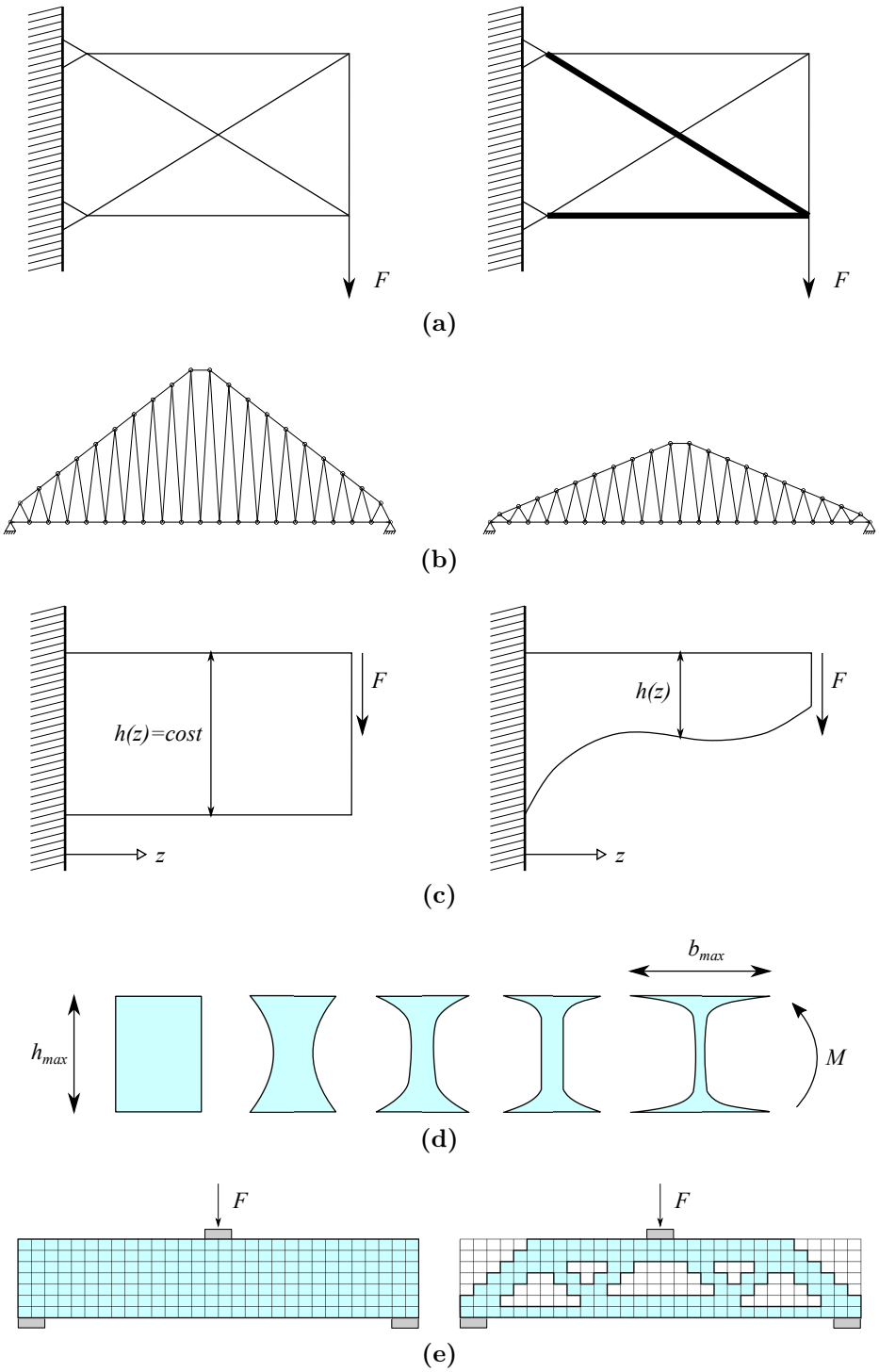



**Figure 1.1:** On the left column, starting design situation are presented; on the right column, optimized design are depicted. (a) Size optimization, image taken by [9]; (b) Shape optimization of a Warren truss beam; (c) Shape optimization with a parametric function $h(z)$, image inspired by [9]; (d) Continuum Topology optimization of a rectangular cross section under a bending moment in elastic stage, image taken by [66]; (e) Topology optmization on a discrete grid, image inspired by [13].

the removal of the less stressed material and, on the contrary, adding material in the most stressed regions [13]. For example, in [66], considering a rectangular section with fixed height under a bending moment load condition in elastic stage, the topology optimization leads to an ideal shape which reminds an I-section profile, as reported in Figure 1.1 (d). Another possible approach to deal with continuum topology optimization is converting it to a discrete topology optimization by meshing the design space and creating cavities or adding material inside the set grid, as depicted in Figure 1.1 (e). This approach is also known as the "ground structure method" [13]. According to [23], the optimization process of a structure does not usually solve simultaneously size, shape and topology optimization. As a matter of fact, as stated in [59], once defined an initial structural layout, it is possible to proceed with a refined shape optimization. As affirmed in [13], size optimization usually is the last one to be performed. As a matter of fact, considering thin shells structures like a dome, the shape strongly influences the internal action distributions and, once the shape is fixed, the size optimization procedure can be adopted to identify the best thickness of the shell [13].

The modern study of the optimization problems dates back to World War II, when the British army recruited a group of mathematicians to face the issue of employing limited resources in an optimal way [43, 64]. The first methods were formalized into the so called *mathematical programming techniques* which now belong to the branch of mathematics called *operations research*. Earlier developments started with Newton and Leibniz with differential calculus, afterwards, other fundamental contributions were given by the greatest mathematicians of the history: Cauchy, Bernoulli, Euler, Lagrange, Weierstrass and others [64]. Later on, new heuristic approaches were developed to solve those problems in which classical methods failed or required extremely high computational costs. These new methodologies were mainly inspired by the Nature, intelligent behaviour and survival strategies of animals. These paradigms also introduced to Artificial Intelligence field (AI) and were mathematically formalized under the category of methods called *meta-heuristic algorithms* to solve optimization problems.

In the present Thesis, in Chapter 2 an overview of the traditional mathematical methods to deal with optimization problems, such as the Lagrange multiplier approach and the mathematical programming methods, is presented. In the last part of this chapter, a brief review of the modern approaches which adopt heuristic to solve complex optimization problems is treated. In Chapter 3, the attention is focused on the meta-heuristic Particle Swarm Optimization (PSO) algorithm which is inspired by the natural swarm intelligent social behaviour of animals such as bird flockings. This intelligent behaviour was adopted to solve optimization problems from its formalization in 1995, but some of the many existing variants presented in this Chapter demonstrate that it is still currently under development in order to improve the performance of the classic algorithm. In the final part of the Chapter, a state of the art on the constraint handling approaches is shown, with particular reference to the most used penalty approaches. After a short digression on Artificial Intelligence (AI) scopes, in Chapter 4 a new non-penalty machine learning constraint handling

approach is presented. The Support Vector Machine (SVM) is combined with PSO as a new method to deal with constrained problems in order to reduce the search space to the feasible region. Finally, in Chapter 5, to test the search performance of the new PSO-SVM algorithm, two numerical literature benchmark problems and two structural examples are presented. The results obtained in the final case studies demonstrate that the PSO-SVM is a new valid alternative which combines the modern AI approaches to solve constrained optimization problems even in the structural optimization field.

# Chapter 2

# Optimization Procedures Review

## 2.1 Single-Objective Optimization Problems

One of the several possible classification for the optimization problems is based on the number of OFs which have to be optimized. A problem where the OF is only one is called single-objective optimization problem. In mathematical terms, as stated in [70], the OF is a function which have to be minimized and which maps a n-dimensional domain to the set of real numbers $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. A single-objective optimization problem can be unconstrained or constrained. The mathematical form for an unconstrained one is: *Find* $\boldsymbol{x}$ *such that*

$$\min_{\boldsymbol{x}\in\Omega}\{f(\boldsymbol{x})\}, \tag{2.1.1}$$

whereas if the problem is constrained it is written as: *Find* $\boldsymbol{x}$ *such that*

$$\begin{aligned} &\min_{\boldsymbol{x}\in\Omega}\{f(\boldsymbol{x})\}, \\ \text{s.t.} \quad &g_q(\boldsymbol{x}) \leq 0 \quad \forall q = 1, ..., n_q, \\ &h_r(\boldsymbol{x}) = 0 \quad \forall r = 1, ..., n_r, \end{aligned} \tag{2.1.2}$$

in which $\boldsymbol{x} = \{x_1, ..., x_j, ..., x_n\}$ is the design variable vector whose components are real numbers, $f(\boldsymbol{x})$ is the objective function (OF) to be minimized and $\Omega$ is a *box-type search space*. For instance, if $[x_j^l, x_j^u]$ is the admissible interval for the $j$-th variable ( $x_j^l$ and $x_j^u$ are its lower and the upper bounds, respectively), then

$$\Omega = [x_1^l, x_1^u] \times ... \times [x_j^l, x_j^u] \times ... \times [x_n^l, x_n^u] \tag{2.1.3}$$

where the symbol $\times$ denotes the Cartesian product between intervals. The constraints of the optimization problem defined by (2.1.2) can be inequalities $g_q(\boldsymbol{x})$ and/or equalities $h_r(\boldsymbol{x})$. Without any loss of generality, all equalities can be easily converted into inequalities. Considering an equality constraint $h_r(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ and considering a certain small tolerance $\varepsilon$ (usually set to $10^{-6}$, [70]), it is possible to transform a equality constraint into a couple of inequalities considering $|h_r(\boldsymbol{x})| \leq \varepsilon$. Therefore, it is possible to consider a problem subjected only to inequality constraints, e.g. $g_p(\boldsymbol{x}) \leq 0$, for $p = 1, ..., n_q, n_{q+1}, ..., n_p$, being $n_p = n_q + 2n_r$. As

reported in [59], in the real-world structural optimization problem usually inequality constraint only are present.
In structural optimization, as affirmed in [64], constraints can be represented as e.g. a physical limitation ("*geometric or side constraints*") or a performance limitation ("*behavior or functional constraints*"). In short, constraints represent a limitation of the design space and divide it into two hyperspace: the feasible region and the unfeasible one. At the end of the optimization process, the optimal solution must respect the constraints and it must lie in the feasible region. As stated in [64], the hyper-surface which divides the feasible region from the unfeasible one is called *constraint surface* and it is usually a "*composite surface*" because related to the combination of different constraints. It is not necessary that all the constraints must contribute to the hyper-surface and, for this reason, it is possible to distinguish active constraints from inactive ones. In mathematical terms, an active constraints is when happen that at the optimum point $\tilde{\boldsymbol{x}}$ the value of the $j$-th inequality constraint become $g_j(\tilde{\boldsymbol{x}}) = 0$ whereas the constraint is called inactive if $g_j(\tilde{\boldsymbol{x}}) < 0$ [64]. A typical example of OF in structural optimization problems is represented by the weight of the structure. In general, it is possible to write the weight as the sum of the weight of all the structural members, which are in total $N_{el}$, under the hypothesis of using homogeneous[1] members:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_{el}} \rho_i V_i(\boldsymbol{x}), \tag{2.1.4}$$

where $\rho_i$ is the density of the $i$-th member and $V_i$ is the volume of each member which directly depends on the design variable vector $\boldsymbol{x}$. Under the assumption of using prismatic elements which have constant area $A_i(\boldsymbol{x})$ along their length $L_i$, it is possible to further simplify the previous OF as:

$$f(\boldsymbol{x}) = \rho \cdot \sum_{i=1}^{N_{el}} A_i L_i. \tag{2.1.5}$$

As one can check, in the OF there are other terms besides the design variables, such as the density in (2.1.4) and (2.1.5). These quantities are usually predetermined and fixed as known data and so they are called *preassigned parameters* in [64].
This kind of OF is not the unique possible but the choice of an appropriate OF depends on the nature of the problem and what is the aim of the problem that an engineer is asked to solve. In civil engineering is typical to look for a minimum cost therefore it is required to use a different OF maybe taking into account not only the construction material costs but also e.g. the labor and maintenance costs [23]. Other times, it is required to find the best mechanical performance for a component and the OF could be different from the previous cases.

Another very interesting structural problem which can be modelled as single-objective optimization problem is the parametric identification problem for the dynamic identification of a structure [62]. In fact, starting from experimental data collected in situ, the problem is to make a reliable non-linear FEM model which exhaustively characterize the dynamic response of the structure under study. In that

[1] Each $i$-th member is made of the same material.

sense, the design vector $\boldsymbol{x}$ is a vector of non-linear model parameters and the OF is a measure of the similarity between the experimental response of the structure $\hat{\boldsymbol{y}}$ and the simulated numerical response of the model $\boldsymbol{y}(\boldsymbol{x})$. As stated in [62] the OF can be assumed as the normalized mean square error which has to be minimized finding the best non-linear parameters vector,

$$f(\boldsymbol{x}) = \frac{100}{N\sigma_y^2}[\hat{\boldsymbol{y}} - \boldsymbol{y}(\boldsymbol{x})]^T[\hat{\boldsymbol{y}} - \boldsymbol{y}(\boldsymbol{x})], \tag{2.1.6}$$

where $N$ is the number of the data point and $\sigma_y^2$ is the variance of the measured response of the structure under study [62].

Based on the nature of the design variables and how they are allowed to vary in the design space, the optimization problems can be divided in two categories. The first category is called *discrete problems* and it is related to the fact that the design variables can assume only discrete values in a finite set of possible values. A typical example of discrete problem is the optimization a steel structure composed by commercial profiles. As a matter of fact, the cross section properties can assume only discrete values respect to the specific dimensions available on the steel profiles commercial tables.
The second category is called *continuous problems* because the design variable can vary continuously in the design space. As reported in [59], in general, it is possible to write that the $i$-th design variable belong to a bounded set which can be composed by or continuous or discrete values:

$$\boldsymbol{x}_i \in X_i \quad \text{for } i = 1, ..., n \tag{2.1.7}$$

For continuous problems, the search space is also called box-type search space because, as expressed in (2.1.3), each design variable is upper and lower bounded, $x_i^l \leq x_i \leq x_i^u$, and it can continuously vary inside its admissible interval. As suggested by [59], when it is possible, discrete problems are treated as continuous ones and at the end of the optimization process a rounding-off procedure is performed in order to assign to the design variables the closest discrete values to the optimal numerical continuous results. This is done because it is more simple to handle with continuous problems respect to discrete ones.

The most used methods to solve optimization problems are distinct in two main groups. The first main group of methods are called, in general, *gradient-based methods* which are also called *deterministic* or *mathematical methods* in [59]. These methods require the information about the gradient and, in general, the OF needs to be regular enough and differentiable to solve the minimization problem. The second main group of methods are called *meta-heuristic methods* and are also called *probabilistic* methods in [59]. These new modern approaches do not require the gradient information and so don't require the OF is regular. They make a certain use of probabilistic mechanism and they are also called Nature-inspired because their derivation takes inspiration by Nature, genetic mechanism, Darwinian evolution theory, behaviour of swarms and flocks, etc.
A detailed overview of all these methodologies is treated in the following of this Thesis.

## 2.2 Multi-Objective Optimization Problems

In analogy with the above, an optimization problem where there are present more than one OFs is called *multi-objective optimization problem.*
In more general terms, in [59] a multi-objective optimization problem (or also known as multicriteria optimization, multiperformance or vector optimization problem [11]) can be formulated as in the following

$$\begin{aligned} &\textit{Find } \boldsymbol{x} = [x_1, ..., x_n]^T \in \Omega \subseteq \mathbb{R}^n \textit{ such that} \\ &\quad \min_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}), \quad \text{with } \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m \\ &\text{s.t.} \quad g_q(\boldsymbol{x}) \leq 0 \qquad \forall q = 1, ..., n_q, \\ &\quad h_r(\boldsymbol{x}) = 0 \qquad \forall r = 1, ..., n_r, \end{aligned} \tag{2.2.1}$$

where the design variables vector $\boldsymbol{x}$ has $n$ dimensions with a box-constraint to each variable ($x_i^l \leq x_i \leq x_i^u$ which are also called side constraints) whereas the vector of the OFs $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]^T$ has $m$ components. According to [19], when $m$ is substantially big, the problem can also be identified as a *many-objective optimization problem.* As already explained before, the equality constraints can be transformed into couples of inequality constraints leading to a formulation of the problem subjected to only inequality constraints. As showed above, in the multi-objective optimization problems usually two Euclidean space are involved [11]: the *design variables space* which is $n$-dimensional and where the coordinate axis are represented by each design variable; the *objective space* (or OFs space) which is $m$-dimensional and where the coordinate axis are represented by each OF. As stated in [11], each point in the design variables space represents a potential solution and, through an evaluation of the OFs, it is mapped to output vectors in the objective space where it is possible to evaluate the quality of the potential solution. The image of the feasible region in the objective space is also called *criterion space* in [59] or attained set. In mathematical terms

$$\boldsymbol{f} : \Omega \longrightarrow \mathbb{R}^m. \tag{2.2.2}$$

As stated in [59], since in real-world problems the OFs are related to contrasting aims which may not be minimized simultaneously, the optimal solution often is not well defined in multi-objective optimization problems. In fact, considering all the OFs together, it may happen that none solution is better than others but the optimality is reached in a sort of trade-off. As affirmed in [11], in the contest of multi-objective optimization problems,

> "*the term optimize means finding such a solution which would give the values of all the objective functions acceptable to the decision maker*".

In order to assess the quality of a certain solution, the evaluation of the OFs is needed. In [11], the OFs can be classified as "commensurable" if they are measured in the same units otherwise they are "non-commensurable". The feasible set is completely ordered in a single-objective optimization problem and between two different solutions it is always possible to establish which one is better through a simple evaluation of the OF [59]. Instead, in a multi-objective optimization problem the feasible

set is only partially ordered and it is not possible to unequivocally compare two possible solutions. For this reason, to evaluate the quality of the solutions it is necessary to adopt the *Pareto order theory* for a minimization problem (also denoted as Edgeworth-Pareto order theory in [11]) which involves binary relations between two candidates solutions. Referring to [19], given two output vectors in the objective space $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^m$, it is said that $\boldsymbol{u}$ *Pareto dominate* $\boldsymbol{v}$ ($\boldsymbol{u} \prec \boldsymbol{v}$) if and only if

$$\begin{aligned} u_i \leq v_i, \quad \forall i = 1, 2, ..., m, \\ u_j < v_j, \quad \exists j = 1, 2, ..., m, \end{aligned} \tag{2.2.3}$$

and in [11] it is said that $\boldsymbol{u}$ *weakly Pareto dominate* $\boldsymbol{v}$ ($\boldsymbol{u} \preceq \boldsymbol{v}$) if only the first condition in (2.2.3) occurs. The vector $\boldsymbol{u}$ is said to *strictly Pareto dominate* $\boldsymbol{v}$ (in symbols $\boldsymbol{u} \prec\prec \boldsymbol{v}$) if and only if [59]

$$u_i < v_i, \quad \forall i = 1, 2, ..., m. \tag{2.2.4}$$

In [59], two vectors $\boldsymbol{u}, \boldsymbol{v}$ are called *incomparable* or indifferent (in symbols $\boldsymbol{u} \prec\succ \boldsymbol{v}$) if does not hold neither $\boldsymbol{u} \preceq \boldsymbol{v}$ nor $\boldsymbol{v} \preceq \boldsymbol{u}$. In [11], in the design space, a vector $\tilde{\boldsymbol{x}}$ is called *Pareto optimal* with respect to $\Omega$ if and only if there no exist another vector $\boldsymbol{x}$ for which $\boldsymbol{v} \in \mathbb{R}^m$, such that $\boldsymbol{v} = F(\boldsymbol{x}) = [f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]$, dominates $\boldsymbol{u} \in \mathbb{R}^m$, such that $\boldsymbol{u} = F(\tilde{\boldsymbol{x}}) = [f_1(\tilde{\boldsymbol{x}}), ..., f_m(\tilde{\boldsymbol{x}})]$. As stated in [64], the Pareto optimal is a feasible solution for which does not exist any other feasible solution that can reduce some OFs without generating a simultaneous rise in at least one of the other OFs. The Pareto optimal set is nothing more than the collection of the output vectors of the all the Pareto optimal points which are also called *non-dominated solutions* in the objective space, in symbols

$$\mathbf{P} = \{\tilde{\boldsymbol{x}} \in \Omega \textit{ such that } \nexists \boldsymbol{x}, \ F(\boldsymbol{x}) \preceq F(\tilde{\boldsymbol{x}})\}, \tag{2.2.5}$$

and the Pareto front is defined as

$$\mathbf{P_F} = \{\boldsymbol{u} = F(\boldsymbol{x}) \textit{ with } \boldsymbol{x} \in \mathbf{P}\}. \tag{2.2.6}$$

As affirmed in [48], the pareto optimal front represent substantially the projection of the Pareto optimal set in the objective space. For a maximization problem the definitions above changes but it is sufficient to reverse the symbols $\preceq$ ($\prec$) with $\succeq$ ($\succ$) and the symbols $\leq$ ($<$) with $\geq$ ($>$). All the previous definition can be considered both in local sense (referring to a neighbourhood) or in a global sense (referring to the entire domain).

According to [59], the points on the Pareto front are "*optimal in the sense that they cannot be improved in one objective without causing deterioration in the other objective*". As depicted in Figure 2.1, an example of the dominance relations are graphically showed in a case with two OFs. Setting the reference vector as $\boldsymbol{u}$, it weakly dominate the vector $\boldsymbol{v}$ ($\boldsymbol{u} \preceq \boldsymbol{v}$) because $f_{1,\boldsymbol{u}} = f_{1,\boldsymbol{v}}$ and $f_{2,\boldsymbol{u}} < f_{2,\boldsymbol{v}}$. Instead, $\boldsymbol{u}$ is strictly dominate the vector $\boldsymbol{t}$ ($\boldsymbol{u} \prec\prec \boldsymbol{t}$) because $f_{i,\boldsymbol{u}} < f_{i,\boldsymbol{t}}\, i = 1, 2$, whereas it is incomparable with $\boldsymbol{w}$ ($\boldsymbol{u} \prec\succ \boldsymbol{w}$) because despite $f_{1,\boldsymbol{u}} < f_{1,\boldsymbol{w}}$, the second component is $f_{2,\boldsymbol{u}} > f_{2,\boldsymbol{w}}$. In fact, due to the fact that the problem require to minimize both $f_1$ and $f_2$ in this last case it is not possible to evaluate which one solution is better only looking to the OF evaluations.

Referring to the Figure 2.2, the top-left figure shows the Pareto front in a minimization problem of two OFs and the top-right figure illustrates the Pareto front in the opposite maximization problem. Despite these two cases, it is often that the OF aim to opposite or conflicting interests and maybe a multi-objective require to maximize some OFs and minimize others, and vice versa. In Figure 2.2, the bottom-left image presents the Pareto front resulting by considering to maximize the OF $f_1$ and to minimize the OF $f_2$, on the contrary the bottom-right image shows a problem which require to minimize $f_1$ and maximize ($f_2$). The decision maker needs to consider the best solution taking into account the Pareto front which represents the trade-offs between the various OFs. As illustrated in the top-left image of the Figure 2.2, as stated in [27], a possible way to help the decision maker is finding the Utopia point (or utopical or ideal vector [11]), which is an ideal point which minimizes all the OFs, and considering the minimum distance, measured in a certain metrics (e.g. the euclidean distance), between this utopical vector and the Pareto front.

As stated in [48, 19], respect to the moment when the decision maker has to point out which is the best optimal trade-off among the different OFs, it is possible to classify the multi-objective optimization problem in three way adding this additional information: *a priori*, *interactive* and *a posterior* methods. In the first one a prior ordering in the objective space is done by the decision maker before performing the optimization algorithm, e.g. giving some weight to each OF and adopting some utility functions. In the a posteriori methods, the partial ordering is taken into account by the Pareto optimal theory and the decision maker gets the information of the Pareto front of the non-dominated solutions after the optimization process. In the interactive (or online or progressive) methods, the decision maker can decide to perform some refinement based on online feedbacks given back to him during the optimization procedure. This latter is substantially a combination of a priori and a posterior methods and, notwithstanding it represents the current most interesting solution, often the multi-objective optimization problems are still dealt with a pos-

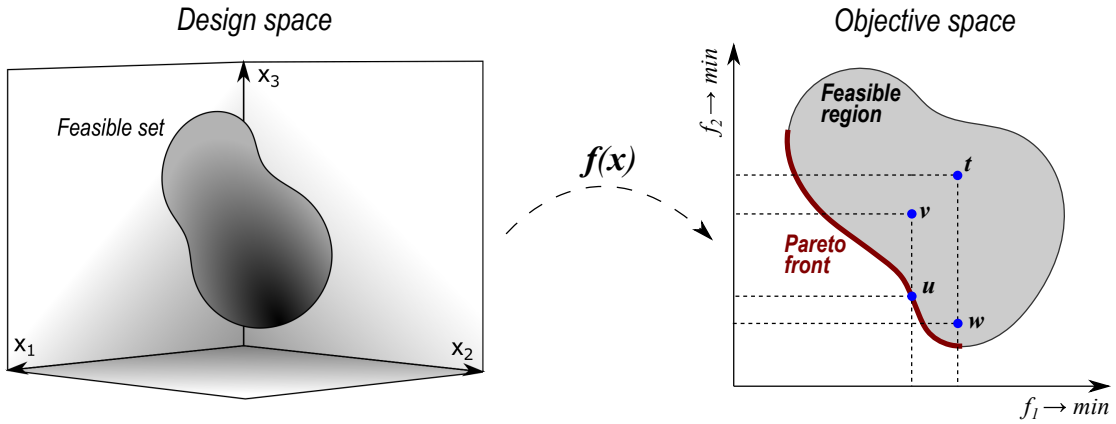



**Figure 2.1:** Multi-objective optimization problem (minimization case): Design space visualization and Objective space considering a three-dimensional design vector and a bi-dimensional OFs vector. The Pareto Front is enlighten in red. The output vectors $\boldsymbol{t}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ are enlightened to show the dominance relations ( $\boldsymbol{u}$ as reference vector): $\boldsymbol{u} \preceq \boldsymbol{v}$, $\boldsymbol{u} \prec\prec \boldsymbol{t}$ and $\boldsymbol{u} \prec\succ \boldsymbol{w}$.

teriori approach [19].

In [64], a simple approach to solve multi-objective optimization problems is to reconduct towards a single-objective optimization problem creating a new OF as a linear combination of the $m$ OFs:

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{m} w_i \cdot f_i(\boldsymbol{x}). \tag{2.2.7}$$

The coefficients $w_i$ can be assumed constant and they play a role of weights of a sort of priority of the different OFs. Therefore, they contribute to give much importance to some OFs respect to others. This solution is also called *scalarization method* in [27] or weighted-sum method or linear weighting method in [59] and it can be identified as a priori method. The weights can be chosen equally or taking into account a sort of rank methods through literature relations. It is important to stress that these weights are arbitrary chosen by the decision maker and they may not necessarily

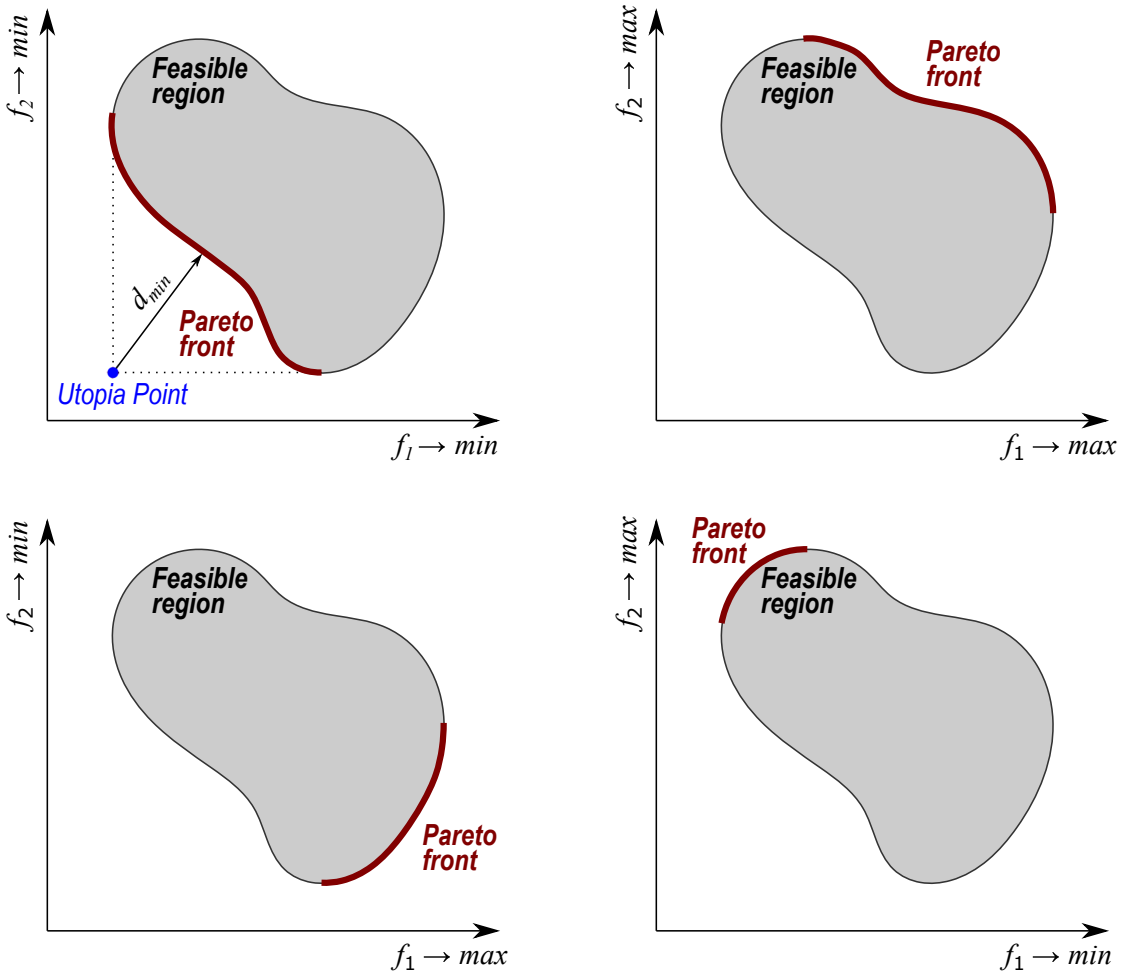



**Figure 2.2:** Image inspired by [48]: graphical representation of the Pareto front in the objective space of some different minimization and maximization multi-objective problems with two OFs.

represent the relative importance of each OFs. The OFs to be minimized are considered with negative sign, whilst the OFs which has to be maximized are considered with positive sign [27]. For furthermore readings about specific techniques which has been developed to face multi-objective optimization problems, one can refer to [11, 19, 27, 48].

The multi-objective optimization problems are encountered very often in real-world problems; a typical example in the structural optimization field can be related to optimization in seismic design. As exposed in [60], it is possible to perform an initial seismic design adopting a constrained single-objective optimization problem taking into account the minimum cost for reinforced concrete (RC) structures and the minimum weight for steel structures. Although this first simplification, considering the entire life-cycle costs which take into account also repairing costs after an earthquake, it transforms the problem in a multi-objective optimization problem. The main difficulties are related to the accurate definition of the restoration costs and, moreover, it should be necessary also considering the uncertainties. As reminded in [60], they are mainly related to the probabilistic nature of the earthquake and to the uncertainty on the material side. This latter is mainly due to the fact that the response of a structure is in non-linear field. Therefore, it is necessary to assess the seismic performances (capacity and demand) at a specific limit state using non-linear approaches. The current methods in practise are to adopt the non-linear static analysis (pushover analysis) or the non-linear dynamic analysis (time history analysis). The early methods to solve the seismic multi-objective optimal design were gradient-based approaches such as linear and quadratic programming, feasible directions and other mathematical methods. As underlined in [60], although their computational efficiency, these approaches required that was possible to explicit the design variables from the OFs and the constraints. To solve this issue the Principle of Virtual Work was adopted looking to closed-form analytical relations or approximated ones. Later, because of research developments in the FEA and in the improvements of computer calculating capacity, the gradient-based methods were abandoned to promote meta-heuristic approaches. One can refer to [60] for furthermore reading of a complete overview about this topic.

## 2.3 Classical Approaches

There are different kind of approaches inside the Operation Research fields: the *mathematical programming* methods involves a set of design variables even subjected to a number of constraints; the *stochastic methods* involves problems with a set of random variables with known probability distribution; *statistical methods* involves experimental data analysis in order to create empirical models which laws best represent the data. The classical approaches are usually related to the gradient-based or mathematical methods. They are called classical methods to distinguish respect to the modern meta-heuristic approaches and because are mathematically "well posed" methods. A optimization problem is nothing less than a minimization

problem of a function and, for this reason, some basic concepts of Mathematical Analysis I are reminded below.
As stated in [5] "*One calls $x_0 \in dom\, f$ a relative (or local) maximum point for $f$ if there is a neighbourhood $I_r(x_0)$ of $x_0$ such that*:

$$\forall x \in I_r(x_0) \cap \operatorname{dom} f, \qquad f(x) \leq f(x_0). \tag{2.3.1}$$

*Then $f(x_0)$ is a relative (or local) maximum of $f$*". If it is valid the same as above not only for the intersection of a neighbourhood with $\operatorname{dom} f$ but for the entire $\operatorname{dom} f$, the point is an absolute maximum of f. Exchanging the $\leq$ symbol with the $\geq$ symobl one gets the definition of both local and global minimum of a map. "*A minimum or maximum point shall be referred to generically as an extremum (point) of $f$*" [5]. Considering a single variable continuous function $y = f(x)$ such that $f : I \subseteq \mathbb{R} \to \mathbb{R}$ on a closed bounded interval $[a, b]$, the Weierstrass Theorem states that this continuous map "*admits minimum and maximum*" [5]. When the map is differentiable, the extremum is also called a stationary point (or critical point) and its first derivative is equal to zero $f'(x_0) = 0$ (Fermat Theorem). In order to find the minimum points of a function in the classic "Qualitative study of a function" one can refer to the theorems of Rolle, Lagrange and Cauchy which are also the basis of the differential calculus. It is also important remind the convexity properties. Considering the same map $f$, which is differentiable in $x_0$, and naming the tangent function as $t(x) = f(x_0) + f'(x_0)(x - x_0)$, "*the map $f$ is convex at $x_0$ if there is a neighbourhood $I_r(x_0) \subseteq dom\, f$ such that*

$$\forall x \in I_r(x_0), \qquad f(x) \geq t(x); \tag{2.3.2}$$

*$f$ strictly convex if $f(x) > t(x)$, $\forall x \neq x_0$*" [5]. It is possible to extend the convexity concept to the concavity considering the symbol $\leq$ and it is possible to define the inflection points where the map change from convexity to concavity and vice versa. Considering an interval $I$, the convexity is also related to the second derivative of $f$ stating that ([5]):

$$f''(x) \geq 0, \quad \forall x \in I \iff f \text{ is convex on } I, \tag{2.3.3}$$

and it is strictly convex considering the symbol $>$. A graphical interpretation of the convexity is that, considering the epigraph[2] of a map, a segment composed by any two point taken in the epigraph or even belonging to the graph of the map, it is entirely contained in this region [59] as depited in Figure 2.3. As stated in [59], a global minimum is also a local minimum but the converse it is true if and only if the function is convex so there exist only one global minimum. According to [64], in general terms, "*a function $f(\boldsymbol{x})$ is convex if the Hessian matrix $H(\boldsymbol{x}) = [\partial^2 f(\boldsymbol{x})/\partial x_i \partial x_j]$ is positive semidefinite*"[3]. The sufficient condition that a stationary point of a function is also a relative minimum is that the Hessian is positive definite,

[2]It is the part of the plane above the graph of a function

[3]As stated in [64], a matrix $\mathbf{A}$ is positive (negative) definite when all the eigenvalues $\lambda$ resulting from the resolution of its secular (or characteristic) polynomial or determinantal equation $\det[\mathbf{A} - \lambda\mathbf{I}] = 0$ are positive (negative). Instead, it is positive (negative) semidefinite if all the eigenvalues are nonnegative (nonpositive).

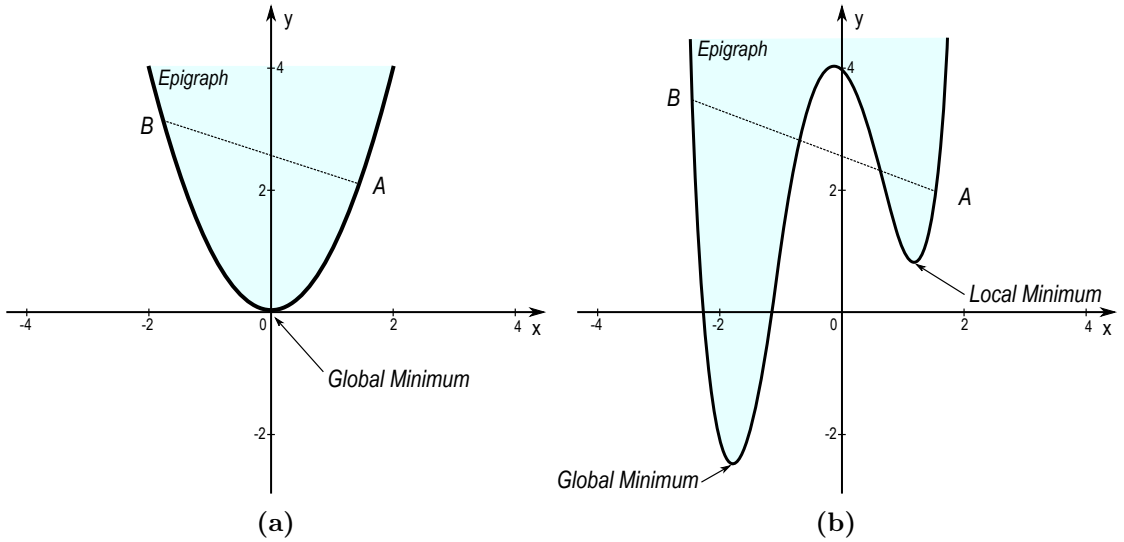



**Figure 2.3:** (a) An example of a convex function $y = x^2$; (b) An example of a non-convex function $y = x^3 - x + \left(2 - x^2\right)^2$; if it was possible to split this function at the local maximum, the two resulting functions, taken alone, would be convex.

whereas to have a relative maximum the Hessian need to be negative definite [64]. According to [59], the mathematical approaches are identified also as *local methods* because they need the study of the second order derivative, which represents the information about the local curvature. In general, a $n$-dimensional set $S$ is a convex set if their elements respect this condition:

$$\text{Taken any } \boldsymbol{x}_1, \boldsymbol{x}_2 \in S, \quad \Rightarrow \quad \boldsymbol{x}(\lambda) = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in S, \quad 0 \leq \lambda \leq 1, \quad (2.3.4)$$

in which $\boldsymbol{x}$ represents a line segment in $n$ dimensions between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. A set which contains only one point is always convex [64]. It is also possible to generalize the convex set definition to the convex combination of $r$ points in the set [64]

$$\boldsymbol{x}(\mu_1, ..., \mu_j, ..., \mu_r) = \mu_1 \boldsymbol{x}_1 + ... + \mu_j \boldsymbol{x}_j + ... + \mu_r \boldsymbol{x}_r, \quad (2.3.5)$$

where each $1 \leq \mu_j \leq 1$ and the condition $\sum_{j=1}^{r} \mu_j = 1$ is respected.

The classical approaches make a strong use of differential calculus methods and this implies that the OF needs to be regular and at least a class $C^2$ function on the domain[4]. Therefore, the mathematical approaches have a very limited field of application in practical optimization problems but they represent a fundamental basis for further and more sophisticated resolution methods [64]. According to [59], the convergence of the mathematical programming methods is faster because they are able to exploit the gradient information finding a sort of a maximum slope direction towards the optimum when they are close to it. Unfortunately, these approaches can very easily being entrapped in local optima and so that they do not ensure the reaching of the global optima. Another important issue to take into account is the very high computational effort in applying them into the structural optimization

[4]The function must be differentiable two times and its derivatives are continuous.

fields in particular because a high rate of operations are already necessarily spent to solve the Finite Element Analysis (FEA) verifying the equilibrium of the structure, as affirmed in [59].

### 2.3.1 Lagrange Multipliers methods

In order to solve single-objective optimization problems with $n$ design variables subjected to $m \leq n$ equality constraints[5] there are several theoretical methods available but only some of them are practically applied. For instance, the *direct substitution* method stated in [64] consist in create a new unconstrained objective function depending only on $m-n$ independent variables. This approach seems extremely simple but it can be applied only for extremely simple problems e.g. when the constraints expressions allow to explicit a design variable alone which can be directly substituted in the OF reducing the dimension of the problem. Another possible approach is the *constrained variation method* which aims to find a closed-form of the total differential of a map $f$ at which the equality constraints are satisfied [64]. The name of this latter is given by the fact that in order to get the total differential $\mathrm{d}f$ it is necessary to consider the arbitrary total variations of its variables $\mathrm{d}x_i$. This method poses the theoretical basis for the most popular Lagrange multiplier method. Considering a simple example taken by [64],

$$\begin{aligned} &\min\ f(x_1, x_2), \\ &\text{s.t. } g(x_1, x_2) = 0, \end{aligned} \tag{2.3.6}$$

the total differental of $f$ is given by considering two arbitrary variations of each variables:

$$\mathrm{d}f = \frac{\partial f}{\partial x_1}\mathrm{d}x_1 + \frac{\partial f}{\partial x_2}\mathrm{d}x_2. \tag{2.3.7}$$

The constraint must be satisfied at the extremum point $(\tilde{x}_1, \tilde{x}_2)$ and the arbitrary variation must be admissible, this means that considering a small total variation from the extremum point, the resulting point must still belong to the constraint function.

$$g(\tilde{x}_1 + \mathrm{d}x_1, \tilde{x}_2 + \mathrm{d}x_2) = 0. \tag{2.3.8}$$

Considering also the Taylor's expansion series it yields to

$$\mathrm{d}g(\tilde{x}_1 + \mathrm{d}x_1, \tilde{x}_2 + \mathrm{d}x_2) = \left[\frac{\partial g}{\partial x_1}\mathrm{d}x_1 + \frac{\partial g}{\partial x_2}\mathrm{d}x_2\right]_{(\tilde{x}_1, \tilde{x}_2)} = 0. \tag{2.3.9}$$

Assuming that $\frac{\partial g}{\partial x_2} \neq 0$ the $\mathrm{d}x_2$ can be expressed in terms of $\mathrm{d}x_1$ and substituting everything inside the (2.3.7) it is possible to get

$$\mathrm{d}f = \left(\frac{\partial f}{\partial x_1} - \frac{\partial g/\partial x_1}{\partial g/\partial x_2}\frac{\partial f}{\partial x_2}\right)\Bigg|_{(\tilde{x}_1, \tilde{x}_2)} \mathrm{d}x_1 = 0. \tag{2.3.10}$$

[5] As stated in [64], if $m$ is greater than $n$, the problem is overconstrained and it is generally impossible to solve.

Since $\mathrm{d}x_1$ is arbitrary, the necessary condition of existence of an extremum point is given by ([64]):

$$\left(\frac{\partial f}{\partial x_1} - \frac{\partial g/\partial x_1}{\partial g/\partial x_2}\frac{\partial f}{\partial x_2}\right)\Bigg|_{(\tilde{x}_1,\tilde{x}_2)} = 0. \tag{2.3.11}$$

From the previous condition, it is possible to define the *Lagrange multiplier* $\lambda$ as

$$\lambda = \left(\frac{\partial f/\partial x_2}{\partial g/\partial x_2}\right)\Bigg|_{(\tilde{x}_1,\tilde{x}_2)} \tag{2.3.12}$$

Substituting (2.3.12) in (2.3.11), it is possible to get

$$\left(\frac{\partial f}{\partial x_1} + \lambda\frac{\partial g}{\partial x_1}\right)\Bigg|_{(\tilde{x}_1,\tilde{x}_2)} = 0, \tag{2.3.13}$$

and reversing the (2.3.12) it is possible to get

$$\left(\frac{\partial f}{\partial x_2} + \lambda\frac{\partial g}{\partial x_2}\right)\Bigg|_{(\tilde{x}_1,\tilde{x}_2)} = 0, \tag{2.3.14}$$

and remembering that the constraint must be satisfied at the extremum point

$$g(x_1, x_2)|_{(\tilde{x}_1,\tilde{x}_2)} = 0, \tag{2.3.15}$$

this three latter relations represent the necessary conditions of the *Lagrange multiplier method* to have an extremum point. Since it was decided that the arbitrary variation was $\mathrm{d}x_1$, performing the same procedures as before but assuming that $\mathrm{d}x_2$ as arbitrary expression, one can check that the necessary conditions for the Lagrange multiplier methods require at least one of the partial derivative of $g$ be non-zero at an extremum point [64]. The Lagrange multiplier method for this problem require to create a Lagrange or Lagrangian function $L$:

$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda \cdot g(x_1, x_2) \tag{2.3.16}$$

and studying its stationary points, thus posing to zero all the partial derivative of $L$ respect to all the variables and the Lagrange multipliers:

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= \frac{\partial f}{\partial x_1}(x_1, x_2) + \lambda \cdot \frac{\partial g}{\partial x_1}(x_1, x_2) = 0, \\
\frac{\partial L}{\partial x_2} &= \frac{\partial f}{\partial x_2}(x_1, x_2) + \lambda \cdot \frac{\partial g}{\partial x_2}(x_1, x_2) = 0, \\
\frac{\partial L}{\partial \lambda} &= g(x_1, x_2) = 0,
\end{aligned} \tag{2.3.17}$$

As stated in [64], in general terms, for a single-objective optimization problem with $n$ variables subjected to $n_r$ equality constraints, with $n \geq n_r$,

$$\begin{aligned}
&\min\ f(\boldsymbol{x}), \\
&\text{s.t. } h_r(\boldsymbol{x}) = 0 \quad \forall r = 1, ..., n_r,
\end{aligned} \tag{2.3.18}$$

considering a Lagrange multiplier $\lambda_r$ for each constraint it is possible to get the Lagrangian function as

$$L(x_1, x_2, ..., x_n, \lambda_1, \lambda_2, ..., \lambda_{n_r}) = f(\boldsymbol{x}) + \sum_{r=1}^{n_r} \lambda_r \cdot h_r(\boldsymbol{x}). \tag{2.3.19}$$

In order to solve the problem it is necessary to study the stationary point of this function performing the first partial derivative respect to each variable of the design vector and respect to each Lagrange multiplier, i.e.

$$\begin{aligned} &\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{r=1}^{n_r} \lambda_r \cdot \frac{\partial h_r}{\partial x_i} = 0, \qquad && i = 1, 2, ..., n, \\ &\frac{\partial L}{\partial \lambda_r} = h_r(\boldsymbol{x}) = 0, && r = 1, 2, ..., n_r. \end{aligned} \tag{2.3.20}$$

The above obtained system have $n+m$ equations in $m+n$ unknowns which are the optimal design vector $\tilde{\boldsymbol{x}}$ and the vector of the Lagrange multipliers $\tilde{\boldsymbol{\lambda}}$. According to [64], the sufficient condition for a relative minimum to exist $\tilde{\boldsymbol{x}}$ is that the following quadratic form $Q$ is positive definite for all admissible variation $\mathrm{d}\boldsymbol{x}$ for which the constraints are satisfied,

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j} \mathrm{d}x_i \mathrm{d}x_j. \tag{2.3.21}$$

In [64], the Lagrange multipliers represent the degree of relaxation or tightness of the OF to the equality constraint at the optimum point, which is also called "*sensitivity or rate of change*". Since they can affect the optimal value of the OF, they are also called *shadow prices*.

Dealing with a single-objective optimization problem subjected to $n_q$ inequality constraints

$$\begin{aligned} &\min\ f(\boldsymbol{x}), \\ \text{s.t.}\quad &g_q(\boldsymbol{x}) \leq 0 \quad \forall q = 1, ..., n_q, \end{aligned} \tag{2.3.22}$$

the approach proposed in [64] is to transform this problem into an equivalent one subjected only to equality constraints introducing non-negative slack variables $\xi_q^2$ obtaining the following problem

$$\begin{aligned} &\min\ f(\boldsymbol{x}), \\ \text{s.t.}\quad &G_q(\boldsymbol{x}, \boldsymbol{\xi}) = g_q(\boldsymbol{x}) + \xi_q^2 = 0 \quad \forall q = 1, ..., n_q. \end{aligned} \tag{2.3.23}$$

Therefore, it is possible to solve it using the Lagrange multiplier rule writing the Lagrangian function as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\xi}) = f(\boldsymbol{x}) + \sum_{q=1}^{n_q} \lambda_q \cdot G_q(\boldsymbol{x}, \boldsymbol{\xi}), \tag{2.3.24}$$

and studying its stationary points obtaining a $n+2n_q$ equations in $n+2n_q$ unknowns which are $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}}$ and $\tilde{\boldsymbol{\xi}}$:

$$\begin{aligned}
&\frac{\partial L}{\partial x_i}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\xi}) = \frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \sum_{q=1}^{n_q} \lambda_q \cdot \frac{\partial g_q}{\partial x_i}(\boldsymbol{x}) = 0, \qquad i = 1, 2, ..., n, \\
&\frac{\partial L}{\partial \lambda_q}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\xi}) = G_q(\boldsymbol{x}, \boldsymbol{\xi}) = g_q(\boldsymbol{x}) + \xi_q^2 = 0, \qquad q = 1, 2, ..., n_q, \\
&\frac{\partial L}{\partial \xi_q}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\xi}) = 2\lambda_q \xi_q = 0, \qquad q = 1, 2, ..., n_q.
\end{aligned} \tag{2.3.25}$$

The second equation of (2.3.25) ensures the respect of the constraints whereas the third equation implies that or $\xi_q = 0$ or $\lambda_q = 0$. When $\lambda_q = 0$ this means that the $q$-th constraint at the optimum is inactive, whilst when $\xi_q = 0$ the $q$-th constraint at the optimum is active. Considering the active or inactive condition, it is possible to separate the constraints in two subset where $J_1$ represent the active set and $J_2$ the inactive one. As stated in [64], is made in order to simplify the first equation of the system (2.3.25) as

$$\frac{\partial L}{\partial x_i}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\xi}) = \frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \sum_{q \in J_1} \lambda_q \cdot \frac{\partial g_q}{\partial x_i}(\boldsymbol{x}) = 0, \qquad i = 1, 2, ..., n, \tag{2.3.26}$$

and it is also possible to rewrite the second equation as

$$\begin{aligned}
g_q(\boldsymbol{x}) &= 0, \qquad q \in J_1, \\
g_q(\boldsymbol{x}) + \xi_q^2 &= 0 = 0, \qquad q \in J_2.
\end{aligned} \tag{2.3.27}$$

obtaining now a system of $n + p + (n_q - p)$ equations in $n + n_q$ unknowns which are $x_i\,(i = 1, 2, ..., n)$, $\lambda_q\,(q \in J_1)$, $\xi_q\,(q \in J_2)$, where $p$ is the number of active constraints. The (2.3.26) can thus be rewritten considering only $p$ constraints as

$$-\nabla f = \lambda_1 \nabla g_1 + \lambda_1 \nabla g_1 + ... + \lambda_p \nabla g_p, \tag{2.3.28}$$

which express that, at the optimum point, the negative gradient of the OF is a linear combination of the gradients of the active constraints weighted by the Lagrange multipliers (the symbol $\nabla$ denotes the gradient operator). As stated in [64], dealing with a minimization problem, all the $\lambda_q$ (with $q \in J_1$) have to be positive and this is related to geometric reasons due to the feasible direction vector (vice versa they have to be negative for a maximization problem).

Summarizing the above results, the conditions that are necessary to be satisfied at a relative minimum of the OF of a constrained minimization problem are called *Kuhn-Tucker conditions*

$$\begin{aligned}
&\frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \sum_{q \in J_1} \lambda_q \cdot \frac{\partial g_q}{\partial x_i}(\boldsymbol{x}) = 0, \qquad i = 1, 2, ..., n, \\
&\lambda_q > 0, \qquad q \in J_1
\end{aligned} \tag{2.3.29}$$

As reported in [64], it is worth noting that during the derivation of these conditions, one fundamental requirement for their applicability is that "*at least one of the Jacobians of the $n_q$ constraints and $n_q$ of the $n + n_q$ design and slack variables be nonzero*". Unfortunately, in general, these conditions are not sufficient to ensure that the solution is a relative minimum but for the *convex programming problems* class they are both necessary and sufficient to ensure the global minimum. As a matter of fact, the convex programming problems are defined as such problems in which both the OF and the constraints are convex functions. The Lagrangian function is in the form of (2.3.23), and it is simple to notice that, since for a minimization problem $\lambda_q \geq 0$ and $\lambda_q \xi_q = 0$, the Lagrangian function is also convex because a linear combination of convex functions. This implies that the Kuhn-Tucker conditions are necessary and sufficient to ensure the unique existence of a global optimum because there are not present neither local minima nor saddle points.
As stated in [64], in general, if the active constraints are not known, the Kuhn-Tucker conditions are written as

$$\begin{aligned} \frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \sum_{q=1}^{n_p} \lambda_q \cdot \frac{\partial g_q}{\partial x_i}(\boldsymbol{x}) = 0, \qquad & i = 1, 2, ..., n, \\ \lambda_q g_q = 0, \qquad & q = 1, 2, ..., n_q, \\ g_q \leq 0, \qquad & q = 1, 2, ..., n_q, \\ \lambda_q \geq 0, \qquad & q = 1, 2, ..., n_q, \end{aligned} \tag{2.3.30}$$

where the second condition comes from the third equation of the system (2.3.25).

When the optimization problem in general involves both equality and inequality constraints such as in the problem statement (2.1.2), the Kuhn-Tucker conditions become

$$\begin{aligned} \nabla f + \sum_{q=1}^{n_q} \lambda_q \nabla g_q - \sum_{r=1}^{n_r} \beta_r \nabla h_r = \mathbf{0}, \\ \lambda_q g_q = 0, \qquad & q = 1, 2, ..., n_q, \\ g_q \leq 0, \qquad & q = 1, 2, ..., n_q, \\ h_r = 0, \qquad & r = 1, 2, ..., n_r, \\ \lambda_q \geq 0, \qquad & q = 1, 2, ..., n_q, \end{aligned} \tag{2.3.31}$$

where $\beta_r$ and $\lambda_q$ are respectivelyt he Lagrange multipliers of the equality and inequality constraints. The optimality is ensured, i.e. there exist $\tilde{\boldsymbol{\lambda}}$ and $\tilde{\boldsymbol{\beta}}$ which satisfy the Kuhn-Tucker conditions, if $\nabla g_q(\tilde{\boldsymbol{x}})$ with $q \in J_1$ and $h_r(\tilde{\boldsymbol{x}})$ with $r = 1, ..., n_r$ are linearly independent (this property is also called *constraint qualification* in [64]). As stated in [15], the second condition is called as Karush-Kuhn-Tucker complementarity condition and it implies that the Lagrange multipliers are $\lambda_q \geq 0$ for active constraints and $\lambda_q = 0$ for inactive ones.

### 2.3.2 Linear, Quadratic and Non-linear Programming

In this section, some mathematical programming techniques are mentioned. For further and deepen readings, one can mainly refer to [64] from which the materials of the following paragraphs are taken.

As stated in [64], *Linear programming* (LP) involves problems in which OF and constraints are linear functions. In general terms, the statement of s LP problem is given in scalar form or matrix form. The scalar form is

$$\begin{aligned} \min\ & f(x_1, ..., x_n) = c_1 x_1 + ... + c_n x_n, \\ \text{s.t. } & a_{11} x_1 + ... + a_{1n} x_n = b_1, \\ & a_{21} x_1 + ... + a_{2n} x_n = b_2, \\ & \qquad \vdots \\ & a_{m1} x_1 + ... + a_{nn} x_n = b_m, \\ & x_1, ..., x_n \geq 0 \end{aligned} \tag{2.3.32}$$

where $c_j$ (cost coefficients), $b_j$ and $a_{ij}$ with $i = 1, ..., m$ and $j = 1, ..., n$ are constants. It is important to notice that in a LP problem the design variables must be non-negative [64]. In fact they usually represents some physical quantities, but if the problem require to deal with also negative dimensions, they can be transformed as the difference of two non-negative variables $x_j = x'_j - x''_j$. As stated in [64], all the possible values of $x_j$, negative sign, zero value or positive sign, is taken into account considering if $x''_j$ is greater, equal or less than $x'_j$.
Moreover, if the problem deal with $q$ inequality constraints, they can be transformed in equality constraints adding non-negative slack variables $x_{n+1}, ..., x_{n+q}$ if the inequality are in the type of "$\leq$", or subtracting non-negative surplus variables otherwise. For instance, considering only one additional inequality constraint in the type of "$\leq$", the transformation is

$$a_{m+1,1} x_1 + ... + a_{m+1,n} x_n \leq b_{m+1}, \quad \longrightarrow \quad a_{m+1,1} x_1 + ... + a_{m+1,n} x_n + x_{n+1} = b_{m+1},$$

whereas considering only one additional inequality constraint in the type of "$\leq$", the transformation is

$$a_{m+1,1} x_1 + ... + a_{m+1,n} x_n \geq b_{m+1}, \quad \longrightarrow \quad a_{m+1,1} x_1 + ... + a_{m+1,n} x_n - x_{n+1} = b_{m+1},$$

According to [64], the LP solution of interest is when $n \leq m$ because there is infinite solution to the problems and it is possible to look for the optimal solution with the minimum OF, whilst if $m = n$ an unique solution exist without any possible optimization and if $m > n$ it means that there are possibly $m - n$ redundant constraints that can be removed. In [64] it is stated that:

> "*A linear programming problem may have (i) a unique and finite optimum solution, (ii) an infinite number of optimal solutions, (iii) an unbounded solution, (iv) no solution, or (v) a unique feasible point*".

Furthermore it is stated that if the LP is correctly formulated, the feasible region is a convex set and the optima occurs at a vertex or point of this feasible polygon. Collecting the various terms of (2.3.32) in matrix and vector form, the matrix formulation is

$$\begin{aligned} \min\ & f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}, \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{2.3.33}$$

with obviously meaning of each term.

As one can check in [6, 64], a typical LP structural problem is the plastic design of a hyperstatic frame (system of Euler-Bernoulli beams) proportionally loaded by concentrated forces, as depicted in the example in Figure 2.4. The solution of the application of the static theorem of plasticity (lower-bound theorem) coincides with a solution of a LP problem [6]. Instead in [64], the Neil-Symonds method (combination of mechanisms methods) is applied. If the energy absorbing capacity ($U$) is greater or at limit equal to the external loads energy ($E$) for each collapse mechanism, the design will be safe. A collapse mechanism is generated when a sufficient number of plastic hinges develops which always starts from the peak of the moment diagram. The aim is to determine the ultimate moment capacity for each collapse mechanism which lead to the minimum weight of the entire structure.
The weight of the structure is equal to the sum of the weights of both beams and columns, and it is proportional to the cross section properties of each element. These properties conditions the ultimate moment capacity, therefore, according to [64], it is possible to assume that the OF is a linear function of the plastic moment capacities $f(M_{p,b}, M_{p,c})$ through a constant factor $a$ which represent the weight per unit length of a member with unitary plastic moment capacity:

$$f(M_{p,b}, M_{p,c}) = \sum_{b,\,c} \text{weight} \approx a \cdot f(M_{p,b}, M_{p,c}), \tag{2.3.34}$$

where $M_{p,b},\ M_{p,c}$ are respectively the plastic ultimate moment of beams and columns composing the structure under consideration.
In [6], a hyperstatic structure is reconducted to an equivalent isostatic structure with the methods of forces, introducing a number $n$ of internal releases (hinges) in order to take into account hyperstatic moment unknowns ($X_j$). The moment along the structure is

$$M(z) = \lambda M^{(0)} + \sum_{j=1}^{n} X_j M^{(j)}, \tag{2.3.35}$$

where $\lambda$ is the static multiplier of the external loads which are all assumed to change proportionally with respect to the same multiplier, $M^{(0)}$ is the moment due to external load in the section at coordinate $z$ and $M^{(j)}$ is the moment in the considered section with unitary hyperstatic unknown. The static theorem requires the respect of the equilibrium between the statically admissible internal actions with external loads. Substantially, the moment along the structure can not violate the plasticity condition which is given by the plastic moment $\pm M_p$ of the section. In general, it is

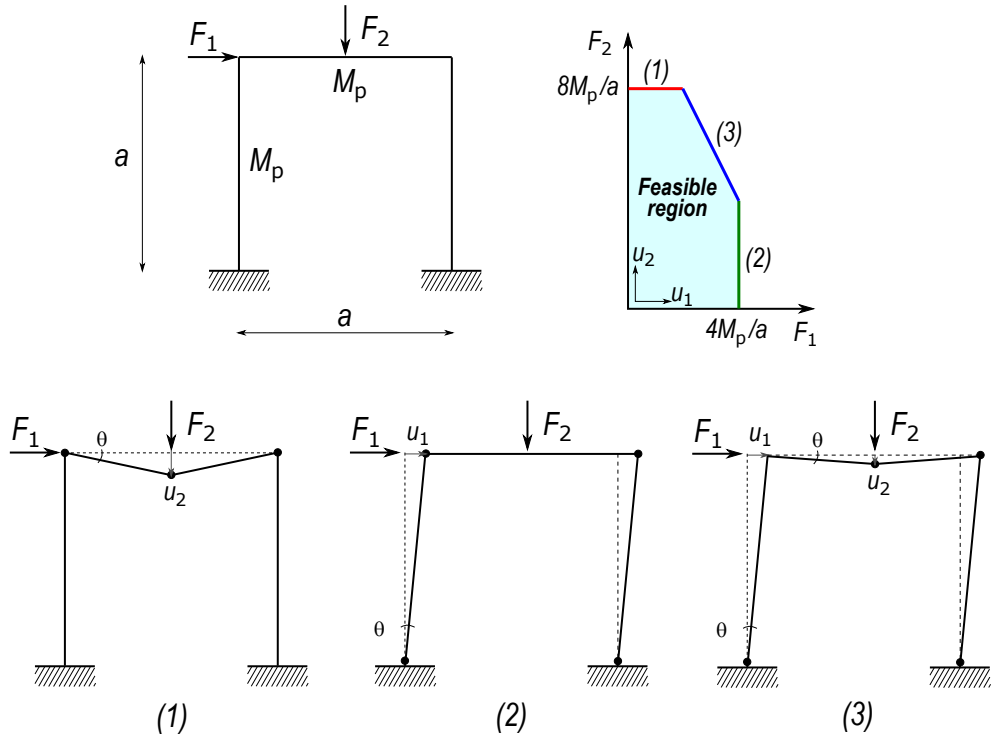



**Figure 2.4:** An example of LP structural problem with a portal frame plastic design taking into account the different collapse mechanism: in (1) the beam mechanism is caused by the assumption $F_2 \gg F_1$; in (2) the lateral mechanism is caused by the assumption $F_1 \gg F_2$; in (3) the combined mechanism is taken into account because $F_1$ and $F_2$ are in the same order of magnitude. In the force space, it is possible to notice that the feasible region (safe region) is inside the limit collapse lines (light blue area).

possible to refer to $m$ critical sections where the moment $M(z)$ is expected to reach the plastic moment value. In practise, considering only the $m$ critical sections the (2.3.35) become

$$M_i = \lambda M_i^{(0)} + \sum_{j=1}^{n} X_j M_i^{(j)}, \quad i = 1, 2, ..., m, \tag{2.3.36}$$

and, considering the static theorem, the (2.3.36) is limited at the plastic conditions and become

$$-M_p \leq M_i \leq +M_p, \quad i = 1, 2, ..., m, \tag{2.3.37}$$

which exactly represent $2m$ constraints of a LP problem in the variables $\lambda$ and $X_j$. With the static theorem, the optimal value is the maximum possible value of the $\lambda$ because it rounds down to the real collapse factor (the red dot in the graph in Figure 2.5).

The solution to LP problems can be obtained in a graphical way only if the design variables are two, but, in general, it can be automatically obtained adopting the most popular *simplex* method by George Dantzig [64]. The traditional method to solve a square system of equation in the form $\boldsymbol{Ax} = \boldsymbol{b}$ is to perform Gauss operations (also known as pivoting operations) in order to get a triangular system which

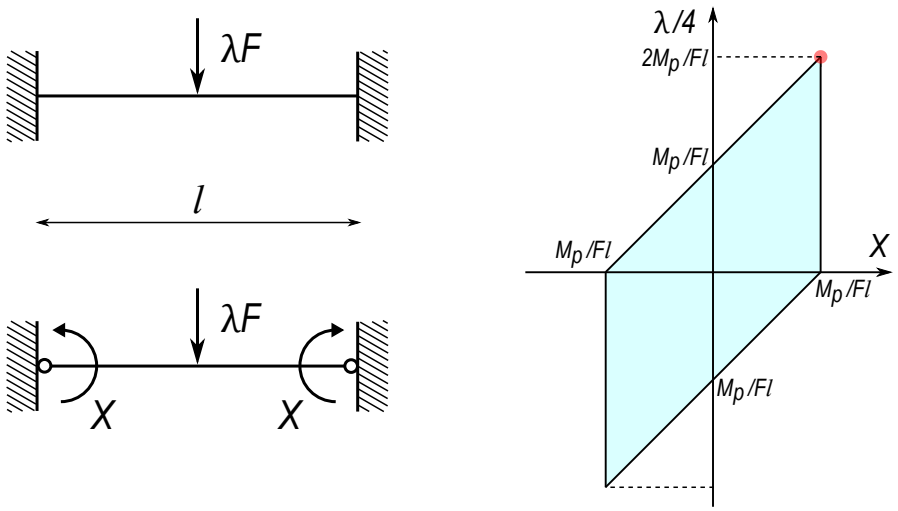



**Figure 2.5:** An example of LP problem with a fixed beam subjected to concentrated external load $F$ which is incrementally risen by the load factor $\lambda$. The structure is only apparently three times hyperstatic. Due to symmetric conditions and considering only flexural problem, it becomes only one time hyperstatic which determine the presence of only one hyperstatic unknown $X$. The red dot of the graph on the right represent the peak value of the load factor which rounds down the real collapse load factor through the static theorem of the theory of plasticity.

can be easily handled to obtain the unique solution. With a rectangular system in the form of (2.3.33), if the system is consistent[6] the pivoting operations can be done in order to get the canonical form of it. From this latter, it is possible to get a basic solution which is a special solution immediately obtained by the system in that canonical form. With additional pivotal operations starting from this canonical form, it is possible to get all the other basic solutions because only one per time is non-zero. It is possible to analyze one solution per time, but many of them could be unfeasible. It is important to notice that this approach become computationally heavy especially when the number of row $m$ and the dimension of the design vector $n$ are very large[7]. Therefore, a more powerful research scheme was implemented thanks to the simplex method introducing non-negative $m$ additional artificial variables to get an auxiliary problem in canonical form from which immediately obtain a basic feasible solution which is then refined and optimized in a second phase of pivoting operations [64]. Despite its advantages, the simplex method requires a quite great amount of computational time and storage memory, therefore, to overcome these issues, a revised simplex method was proposed later. It requires the definition of a dual form of the LP problem which is characterized by the transformation of a minimization problem in a simpler maximization problem which has the same optimal feasible solutions, if they exist. Later other techniques were proposed referring to specific classes of problems or in order to make more efficient the algorithm. For

[6] In [64], it means that the rectangular system have less row than column and it admits at least one solution.

[7] As affirmed in [64], the number of basic solutions to be inspected is equal to the binomial coefficient $\binom{n}{m}$.

instance for special problems which have wide dimensions and a great number of constraints the decomposition principle by Dantzing and Wolfe [64] can be adopted. This allows to split the original problems in a number of sub-problems which are solved almost independently. For furthermore reading about LP methods, one can refer to [64].

According to [64], *quadratic programming problem* (QP) is an example of non-linear programming which is characterized to have a quadratic OF subjected to linear constraints:

$$\begin{aligned} &\min\ f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{D}\boldsymbol{x}, \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ &\quad \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{2.3.38}$$

where $\boldsymbol{x}$ is the $n \times 1$ design vector, $\boldsymbol{c}$ is a $n \times 1$ vector, $b$ is the constant terms $m \times 1$ vector,$\boldsymbol{A}$ is the coefficients' $m \times n$ matrix and $\boldsymbol{D}$ is a $n \times n$ square symmetric positive definite matrix which denotes the quadratic part of the OF. If this part disappear, e.g. $\boldsymbol{D} = \boldsymbol{0}$, it becomes again a LP problem.
To solve the QP in the form (2.3.38), it is possible to adopt the Lagrange multiplier rule transforming the inequality constraints in equality constraints introducing slack variables and surplus variables as showed before in (2.3.23). Denoting $\boldsymbol{A}_i$ as the column vectors of the matrix $\boldsymbol{A}$, each inequality is converted into an equality introducing slack variables $s_i^2$

$$\boldsymbol{A}_i^T\boldsymbol{x} + s_i^2 = b_i, \quad i = 1, ..., m. \tag{2.3.39}$$

The non-negative condition on the design variables is written introducing some surplus variables $t_j^2$

$$-x_j + t_j^2 = 0, \quad j = 1, ..., n. \tag{2.3.40}$$

The Lagrangian function is then given by

$$L(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{D}\boldsymbol{x} + \sum_{i=1}^{m}\lambda_i(\boldsymbol{A}_i^T\boldsymbol{x} + s_i^2 - b_i) + \sum_{j=1}^{n}\theta_j(-x_j + t_j^2) \tag{2.3.41}$$

In order to find a solution, it is necessary to study the stationaryness of the Lagrangian function respect each variable from which it depends on:

$$\frac{\partial L}{\partial x_j} = c_j + \sum_{i=1}^{n} d_{ij}x_i + \sum_{i=1}^{m}\lambda_i a_{ij} - \theta_j = 0, \qquad j = 1, ..., n, \tag{2.3.42}$$

$$\frac{\partial L}{\partial s_i} = 2\lambda_i s_i = 0, \qquad i = 1, ..., m, \tag{2.3.43}$$

$$\frac{\partial L}{\partial t_j} = 2\theta_j t_j = 0 \qquad j = 1, ..., n, \tag{2.3.44}$$

$$\frac{\partial L}{\partial \lambda_i} = \boldsymbol{A}_i^T\boldsymbol{x} + s_i^2 - b_i = 0 \qquad i = 1, ..., m, \tag{2.3.45}$$

$$\frac{\partial L}{\partial \theta_j} = -x_j + t_j^2 = 0 \qquad j = 1, ..., n, \tag{2.3.46}$$

Recalling the slack variables as $Y_i = s_i^2 \geq 0$, the equation (2.3.45) become

$$\boldsymbol{A}_i^T \boldsymbol{x} - b_i = -s_i^2 = -Y_i, \tag{2.3.47}$$

and multiplying the (2.3.43) by $s_i$ and the (2.3.44) by $t_j$ they become

$$\lambda_i s_i^2 = \lambda_i Y_i = 0, \tag{2.3.48}$$
$$\theta_j t_j^2 = 0. \tag{2.3.49}$$

Combining now the equations (2.3.47) with (2.3.48) and (2.3.46) with (2.3.49), two new conditions are obtained

$$\lambda_i(\boldsymbol{A}_i^T \boldsymbol{x} - b_i) = 0, \tag{2.3.50}$$
$$\theta_j x_j = 0. \tag{2.3.51}$$

As stated in [64], in the end, the solution of the original QP problem can be obtained by solving the following necessary conditions as a system of $2(n+m)$ equations in $2(n+m)$ non-negative unknowns $\lambda_i, \theta_j, x_j, Y_i$:

$$c_j + \sum_{i=1}^{n} d_{ij} x_i + \sum_{i=1}^{m} \lambda_i a_{ij} - \theta_j = 0, \quad j = 1, ..., n, \tag{2.3.52}$$
$$\boldsymbol{A}_i^T \boldsymbol{x} - b_i = -Y_i, \qquad i = 1, ..., m, \tag{2.3.53}$$
$$x_j \geq 0 \qquad j = 1, ..., n, \tag{2.3.54}$$
$$Y_i \geq 0 \qquad i = 1, ..., m, \tag{2.3.55}$$
$$\lambda_i \geq 0 \qquad i = 1, ..., m, \tag{2.3.56}$$
$$\theta_j \geq 0 \qquad j = 1, ..., n, \tag{2.3.57}$$
$$\lambda_i Y_i = 0 \qquad i = 1, ..., m, \tag{2.3.58}$$
$$\theta_j x_j = 0 \qquad j = 1, ..., n. \tag{2.3.59}$$

As noticed in [64], the equations (2.3.52), (2.3.53) are linear function of the non-negative unknowns, but it is not true for the last two equations (2.3.58) and (2.3.59). Since the $\boldsymbol{D}$ matrix were defined as symmetric positive definite matrix, this implies that the OF is strictly convex and, therefore, the solution is unique because, by definition, any local minimum is also a global minimum. As affirmed by [64], this kind of problem can also be reconducted to solving a LP with simplex methods adding non-negative artificial variables but imposing at any time the respect of the non-linear conditions (2.3.58) and (2.3.59).

Usually, in real-world engineering problems, the OF and the constraints are complicated to allow solving them analytically because they need to manipulate and explicit the functions in terms of the design variables. For instance, in [64], the problem of minimum weight of a planar truss simply supported beam is showed

adopting an OF in the form of (2.1.5). For truss problems, usually the deformability problems conditioned the design, therefore the constraints are set as limiting the maximum displacement of each node $u_j$ at a certain limit $\delta$ in each direction considered $g(\boldsymbol{x}) = |u_j| - u_{lim} \leq 0$ with $j = 1, ..., n$ number of nodes. Reminding that, in the matrix automatic computation of the structures, the problem is governed by $\boldsymbol{K\delta} = \boldsymbol{F}$, where the $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{\delta}$ is the vector of the displacements and $\boldsymbol{F}$ is the vector of the forces, it is not possible to explicit the functions to solve it analytically. For this reason, the *non-linear programming* (NLP) methods are usually configured as numerical iterative approaches which exploit the gradient information and local curvature information in order to find a search direction towards the optima [59]. As expressed in [64], in general, the NLP methods start with a trial solution $\boldsymbol{x}_1$ and look for a search direction $\boldsymbol{S}_i$ with some criteria. After defined the step length of search $\lambda_i^*$, the point is moved in that direction with that step obtaining a new approximation which is called also "efficiency of the optimization method",

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda_i^* \boldsymbol{S}_i. \tag{2.3.60}$$

At that point, the optimum check is performed in order to understand if the procedure has to stop or to start again. Since the OF depends on $\boldsymbol{x}$ but it is approximated to $\boldsymbol{x}_{i+1}$, the OF can be rewritten as a problem to determine the step $\lambda_i$ which minimize the OF. Early search methods were developed to solve uni-dimensional problems (OF is only function of $\lambda_i$) such as eliminations methods which assumes a simpler but not efficient fixed step length or an accelerated step length. *Exhaustive method* is called simultaneous methods and it is usually applied when it is known that the optimal is contained in a finite interval considering for example a starting solution and a level of uncertainty around this value. More efficient sequential searches are usually adopted such as the dichotomous approach which split the uncertain set at each time. This reduces to the bisection method if it halve the interval at each iteration. Other splitting procedures are available such as Fibonacci search or Golden search. *Interpolation methods* try to approximate the gradient of a uni-dimensional OF respect to $\lambda_i^*$ variable in order to determine the best step $\lambda_i^*$. Similarly to what happens in FEM, the shape functions adopted to perform the interpolations can be roof functions (piecewise linear functions), but usually best results are obtained with quadratic interpolation and cubic interpolation (spline). The *direct root methods* foresees to approximate the derivative of the OF as Taylor's expansion series stopped at least at the second-order and search for the root of the resulting expression. The Newton's method, better known as Newton-Raphson, operates in an iterative way and corresponds to an equivalent quadratic approximation of the function. The Newton-Raphson method and also its main variant called modified Newton-Raphson are often use when, in combination with FEA, it is necessary to take into account the non-linearity stage. In fact, after the elastic stage, the material damaging is taken into account updating the stiffness matrix which affects the forces distribution along the structure. If the second-order derivative is non-zero, the Newton method has a very fast convergence which is called quadratic convergence. When it is not possible to easily differentiate the function or it is not available in closed form, the quasi-Newton method is more suitable because approximate the function using the finite difference formulas. In the secant method, the

first derivative is approximated between two points in a linear secant way, which implies that the function is approximated as a second-order function. In [64] it is stated that when interpolation polynomials are not representative of the behaviour of the OF, the Fibonacci and Golden search will probably operate better, whilst when the interval of uncertain is not available quadratic and quasi-Newton methods it is expected to be more effective. In the cases when the fist derivatives is available, cubic interpolation or secant methods should be the most suitable.

Despite the unconstrained problems rarely appear in real-life engineering problems, the unconstrained search methods represents the basis for many mathematical programming constrained search. Some complex problems can anyway be reconducted to unconstrained search such as the study of displacement response of a structure under any load condition. As a matter of fact, it can be solved by the general method of minimizing the potential energy. This is a useful approach for example adopted in the buckling analysis. As already said before, the minimum of a function is situated in a stationary point ($\partial f/\partial x_i|_{\boldsymbol{x}^*} = 0, \forall i = 1, ..., n$) and in general it is guaranteed if the Hessian matrix is positive definite. However, if the OF is not differentiable it is not possible to apply this criterion, therefore the *direct search methods*, also known as *nongradient methods*, were developed. They are also called zeroth-order methods because they only require the evaluation of the OF and do not require any condition on its differentiability. Since all of the numerical approaches perform an iterative sequential procedure, the evaluation of the success of this procedure is performed looking at the rate of convergence. It represents a ratio between the norm of the absolute errors measured in two consecutive iteration respect to the real optima point [64]:

$$\frac{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_i - \boldsymbol{x}^*\|^p} \leq T, \tag{2.3.61}$$

where the norm of a $n$-dimensional vector is $\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}$, $T \geq 0$ is a certain tolerance and $p \geq 1$ affects the order of the denominator. If $p = 1$ and $0 \leq T \leq 1$ the algorithm is defined as linearly convergent, whereas if $p = 2$ the convergence is more fast and it is denoted as quadratically convergent. This latter means that a quadratic function is minimized in a finite number of operation, in particular in $n$ steps or less [64].
The *random search method* exploits random numbers generators to explore the box search space assigning random values to each design variable. Through an evaluation of the OF, it is possible to establish which point is the best found up to this iteration. This is not an efficient approach but in order to improve this system the *random walk method* was formulated. It assumes the approximation at $i$-th iteration as (2.3.60) but fixing the step length $\tilde{\lambda}_i$ to a prescribed constant value and defining a unit random vector $\boldsymbol{u}_i$ to fix a new random search direction at each iteration:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \tilde{\lambda}_i \boldsymbol{u}_i. \tag{2.3.62}$$

This system was further improved in the *random walk method with direction exploitation* which further implements a uni-dimensional search to find the best optimal step

$\lambda_i^*$. The *grid search methods* create an ideal mesh in the design space and evaluate the OF at each node establishing which one is the best. Since the great number of OF evaluations needed, this procedure is not very efficient, especially with dense grids. In the *univariate method*, the problem is reconducted to a one-dimensional minimization because at each iteration only one variable in turn is changed. This procedure does not converge so rapidly and another drawback is the tendency to oscillate when it is close to the optimum. In *pattern search methods* the allowed direction is only that direction parallel to coordinate axis but also here the convergence may be slow. An extension of that method is the *Powell's method* which adopt the quadratic approximation of a non-linear function and exploit the concept of the conjugate directions[8]. In [64] it is demonstrated that that method is quadratically convergent. The simplex is a geometric figure which in the plane corresponds to a triangle, in the three-dimensional space correspond to a tetrahedron and, in general, it is formed by a set of $n+1$ points. It is said to be regular if the points which form it are equidistant. The simplex method substantially adopts this figure which can move in an iterative process in the design hyperspace and the OF is evaluated at the $n+1$ vertices of this figure. The method adopts some geometrical rules as the reflection property, expansion property and the contraction property to make a robust search avoiding the stall in a non-optimal solution.
The *indirect search or descent methods* require the differentiability of the OF at the fist-order or sometimes also at the second-order, besides the OF evaluations. For this reason, they are also called *gradient methods* of first-order or second-order methods if they respectively require the fist order or the second order derivative. The most important local property of the gradient of a $n$ dimensional function is that it represents the direction where the function increases at the maximum rate and it is called direction of steepest ascent. On the contrary, considering the negative gradient, this becomes the direction of steepest descent. In order to reduce the numerical errors and bad conditioning of the Hessian matrix, it can be useful to perform a scaling procedure in order to rescale the design variables (reducing to a diagonal matrix) and also normalizing them (reducing to a unit matrix) through the formulations suggested in [64]. The formulations are given for quadratic functions but it is usual in gradient-based approaches to locally approximate the OF with a second-order Taylor's expansion series.
The evaluation of the gradient by analytical calculation of all the partial derivatives respect to all design variables may not always be easy. It is possible that it requires large computations or it may also be impossible because the gradient is not always defined. In this latter case, one has to abandon the gradient methods adopting a direct search method as above. In the other cases, it is possible to use the forward finite-difference formula to approximate the gradient

$$\left.\frac{\partial f}{\partial x_i}\right|_{\boldsymbol{x}_m} \simeq \frac{f(\boldsymbol{x}_m + \Delta x_i \boldsymbol{u}_i) - f(\boldsymbol{x})}{\Delta x_i}, \quad i = 1, ..., n, \tag{2.3.63}$$

where $\boldsymbol{x}_m$ is called base point, $\Delta x_i$ is the step length measured forward respect to

[8] Considering a $n \times n$ matrix $\boldsymbol{A}$, a set of $n$ vectors which define a direction $\{\boldsymbol{s}_i\}$ is A-conjugate if $\boldsymbol{s}_i^T \boldsymbol{A} \boldsymbol{s}_j = 0$, $\forall i \neq j$, with $i, j = 1, ..., n$. The orthogonal direction definition is contained in this definition because it is only a special case i.e. when $\boldsymbol{A} = \boldsymbol{I}$.

the base point and $\boldsymbol{u}_i$ is a vector which has the only the $i$-th component non-zero and equal to one. As enlighten in [64], $n$ OF evaluations are needed to estimate the gradient of the OF in the considered point $\boldsymbol{x}_m$. In order to get better results, it is possible to estimate the gradient using the central finite-difference formula but taking into account two additional OF evaluation for each partial derivative:

$$\left.\frac{\partial f}{\partial x_i}\right|_{\boldsymbol{x}_m} \simeq \frac{f(\boldsymbol{x}_m + \Delta x_i \boldsymbol{u}_i) - f(\boldsymbol{x}_m - \Delta x_i \boldsymbol{u}_i)}{2\Delta x_i}, \quad i = 1, ..., n. \tag{2.3.64}$$

The step length $\Delta x_i$ may be chosen with care because, if it is too small, numerical rounding-off errors may prevail, on the contrary, if it is chosen too large, the truncation errors will prevail.
The most ancient gradient method dates back to 1847 when Cauchy formulated the *steepest descendent method*. Starting from an arbitrary point $\boldsymbol{x_1}$ at iteration $i = 1$, an iterative procedure moves the point along the steepest descendent direction toward the minimum of the OF. The search direction $\boldsymbol{s}_i$ is

$$\boldsymbol{s}_i = -\nabla f_i = -\nabla f(\boldsymbol{x}_i). \tag{2.3.65}$$

The optimal step length $\lambda_i^*$ is determined in the direction $\boldsymbol{s}_i$ through one of the previous uni-dimensional search procedures. The value of the approximated point in the next iteration is equal to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda_i^* \boldsymbol{s}_i = \boldsymbol{x}_i - \lambda_i^* \nabla f_i. \tag{2.3.66}$$

The convergence criteria can set in three different ways: as in (2.3.67), when the absolute value of the OF value in two consecutive iteration is smaller than a positive small constant $\varepsilon_1$; as in (2.3.68), when the partial derivatives become smaller than a positive small constant $\varepsilon_2$; as in (2.3.69), when the change in the design variables in two consecutive iteration become smaller than a positive small constant $\varepsilon_3$.

$$\left|\frac{f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}_i)}{f(\boldsymbol{x}_i)}\right| \leq \varepsilon_1 \tag{2.3.67}$$

$$\left|\frac{\partial f}{\partial x_i}\right| \leq \varepsilon_2 \tag{2.3.68}$$

$$|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i| \leq \varepsilon_3 \tag{2.3.69}$$

Notwithstanding its general good behaviour, since the gradient steepest direction is a local property, its convergence is affected by the initial solution.
The *conjugate gradient method* or also know as Fletcher-Reeves method, try to improve the steepest descendent method manipulating it to make it quadratically convergent using the conjugate direction search. It is better than the cauchy methods because it can successfully deals with ill-conditioned quadratic problems but it is less efficient than the following method.
The *Newton's method* can be applied also to multi-variable function problems and it is based on the quadratic approximation of the OF through Taylor's series expansion centred in $\boldsymbol{x}_i$,

$$f(\boldsymbol{x}) \simeq f(\boldsymbol{x}_i) + \nabla f_i^T (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^T \boldsymbol{H}_i (\boldsymbol{x} - \boldsymbol{x}_i), \tag{2.3.70}$$

where $\boldsymbol{H}_i = \boldsymbol{H}|_{\boldsymbol{x}_i}$ is the Hessian matrix evaluated in $\boldsymbol{x}_i$. Since this method require the Hessian matrix, the Newton's method is classified as a second-order method. The necessary condition to get a stationary point is to set the partial derivative of (2.3.70) to zero:

$$\frac{\partial f(\boldsymbol{x})}{\partial x_j} = 0, \quad j = 1, ..., n. \tag{2.3.71}$$

The two previous equations lead to the following approximated stationariness condition

$$\nabla f = \nabla f_i + \boldsymbol{H}_i(\boldsymbol{x} - \boldsymbol{x}_i) = 0. \tag{2.3.72}$$

If the Hessian matrix is not singular, setting $\boldsymbol{x} = \boldsymbol{x}_{i+1}$, the improved approximation at the next iteration is obtained as

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \boldsymbol{H}_i^{-1}\nabla f_i. \tag{2.3.73}$$

As stated in [64], since the higher order terms has been neglected in (2.3.70), the (2.3.73) has to be adopted iteratively to get the optimal solution $\tilde{\boldsymbol{x}}$. If the OF is not a quadratic function it is possible that the method may diverge or converge to saddle or maxima points. To avoid this, in [64], a modification of (2.3.73) is proposed taking into account the optimal step length $\lambda_i^*$:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \lambda_i^* \boldsymbol{H}_i^{-1}\nabla f_i. \tag{2.3.74}$$

The main drawbacks of this procedure is that it need to compute and store the Hessian matrix, then it require its inversion at each step which lead to huge computational effort.
The *Marquardt method* tries to combine the advantages of both the steepest descended method and the Newton's methods by a modification of the diagonal of the Hessian matrix. The modification is done adding positive constants $\alpha_i$ which values is tuned in order to have the resulting Hessian $\tilde{\boldsymbol{H}}_i$ positive definite:

$$\tilde{\boldsymbol{H}}_i = \boldsymbol{H}_i + \alpha_i \boldsymbol{I}. \tag{2.3.75}$$

As affirmed in [64], if that constants are sufficiently large, in the order of the $10^4$, this method reduces to the steepest descendent method because only the substantially diagonal dominate

$$\tilde{\boldsymbol{H}}_i^{-1} = (\boldsymbol{H}_i + \alpha_i \boldsymbol{I})^{-1} \approx (\alpha_i \boldsymbol{I})^{-1} = \frac{1}{\alpha_i}\boldsymbol{I}. \tag{2.3.76}$$

Usually, the constant values are large at the first iterations and they gradually reduce to zero toward the end of the algorithm, therefore passing from a steepest methods at the beginning to a Newton's method at the end. This method can also be easily adapted to take into account the optimal step length $\lambda_i^*$ obtained by a uni-dimensional search procedure.
Another important class of gradient-based formulation is the *quasi-Newton methods*. In these procedures, the approximation is always the same in (2.3.73), but this time the Hessian matrix is approximated by another matrix $\boldsymbol{H}_i \approx \boldsymbol{A}_i$ as well as its

inverse matrix $\boldsymbol{H}_i^{-1} \approx \boldsymbol{B}_i$ using only the first partial derivatives of the OF. At each iteration it is necessary to follow update rules of the approximated matrix in order to guarantee the positive definiteness:

$$\boldsymbol{B}_{i+1} = \boldsymbol{B}_i + \boldsymbol{\Delta B}_i. \tag{2.3.77}$$

Respect to the formulation proposed in the literature for the correction matrix $\boldsymbol{\Delta B}_i$, the most popular quasi-Netwon methods are denoted as *Davidon–Fletcher–Powell* (DFP) and *Broydon–Fletcher–Goldfarb–Shanno* (BFGS). This latter iteratively updates the Hessian and not its inverse which is approximated (indirect update method). As stated in [64], it is numerically proved that BFGS is superior because it is not affected by errors that could occur in the optimal step length $\lambda_i^*$. In short, the BFGS can be considered a quasi-Newton, conjugate gradient and variable metric method. As underlined by [64], in practical applications, it is possible that $\boldsymbol{B}_i$ become indefinite or singular; in that case, some periodical resetting of the matrix to the identity matrix $\boldsymbol{I}$ has to be performed to improve the algorithm.

In order to deal with constrained non-linear problems, other specific mathematical approaches were developed and they can be classified in two main categories: the *direct methods* and the *indirect methods*. In [64] a series of cases of how the presence of constraints can affect the solution of the problem is presented. If the optimum is inside the feasible region, the presence of the constraints may not affect the results. In this case, considering a box-type search space, an unconstrained search will give always the same solution of the constrained problem. Hence, the necessary conditions about the stationariness of the gradient of the OF and the positive definition of the Hessian matrix are also sufficient to solve the problem. In many other cases, the presence of the constraints lead to reduce the feasible space and the optimum can be found on the edge of the feasible region. In that case, the necessary conditions are the Kuhn-Tucker conditions which implies that "*the negative gradient of the OF is a positive linear combination of the gradients of the active constraints*". It is also possible that, due to the fact that the constraints cut the original feasible region, their presence can induce more that one local minimum points which do not exist before.

The *direct methods* directly take into account the presence of the constraints. The *random search method* randomly extract trial design vector and verify the satisfaction of the constraints. If it violated any constraint, the random generation continues, otherwise the solution is stored if the OF is lower than the previous stored solution. From this general layout, many variants were proposed, e.g. combining a feasible directions methods in order to improve the search after randomly finding a feasible solution. Despite its simplicity, it is not efficient but it is useful to find a good starting solution for more sophisticated methods.
The *complex method* by Box in 1965, started as an extension of the simplex method to the constrained problems. Unfortunately, this approach is not able to handle equality constraints. Similarly to the simplex method, the complex method adopts a geometric figure composed of $k \geq n + 1$ vertices under the assumption to start with vertex as an initial feasible point. The other $k - 1$ points are generated ran-

domly and, for each point, the satisfaction of the constraints is performed. If a point $\boldsymbol{x}_j$ violates any constraint, it is iteratively moved halfway toward the centroid of the points already placed in the feasible region until it becomes also feasible. The centroid is defined as

$$\boldsymbol{x}_0 = \frac{1}{j-1}\sum_{l=1}^{j-1}\boldsymbol{x}_l. \tag{2.3.78}$$

After placing all the vertices, the OF is evaluated for all of them establishing the best and the worst. For this latter, which is denoted as $\boldsymbol{x}_h$, in analogy with the simplex method, the reflection procedure is performed respect to the centroid $\boldsymbol{x}_0$ of all vertices except that point

$$\boldsymbol{x}_0 = \frac{1}{k-1}\sum_{l=1}^{k}\boldsymbol{x}_l \cdot (\boldsymbol{x}_l \neq \boldsymbol{x}_h). \tag{2.3.79}$$

For the new obtained point the feasibility is tested and also the OF value. If it is feasible and it has a lower OF, the point is replaced, whereas its distance is iteratively halved moving closer to the centroid until it becomes feasible or the distance is lower a certain tolerance. In that case, the original point $\boldsymbol{x}_h$ is left in its original position and the entire reflection procedure is performed for the second worst vertices (second point with an highest value of the OF). The convergence criteria can be set in two way: when the complex shrink too much and the distance among the vertices become smaller than a positive small constant $\varepsilon_1$; when the standard deviation of the OF of all the vertices become lower than a positive small constant $\varepsilon_2$

$$\sqrt{\frac{1}{k}\sum_{j=1}^{k}[f(\boldsymbol{x}) - f(\boldsymbol{x}_j)]^2} \leq \varepsilon_2. \tag{2.3.80}$$

Box suggest to adopt a large number of vertices at least $k \simeq 2n$ in order to avoid that the complex collapse when it meets the first constraint.
The *sequential linear programming method* (SLP) by Cheney, Goldstein and Kelly tries to solve constrained non-linear problem by solving a number of LP sub-problems by adopting an approximation at the first order of the Taylor's expansion series of both the OF and the constraints. Starting from the initial optima $\boldsymbol{x}_i$ considering only the box-search space (side constraints for the design variables), the next LP problem can be solved considering a linearization of the constraints in the point $\boldsymbol{x}_i$. The LP problem can be treated for example with the simplex method to find a new optima $\boldsymbol{x}_{i+1}$. This latter solution becomes the starting point for the next sub-problem which require a further linearization centred, this time, in $\boldsymbol{x}_{i+1}$. Due to the linearization, for most of the time it is possible that the temporary optimal solutions which lie outside the feasible space and only at the end of the process it become feasible (especially with a convex programming [64]). The SLP is also called cutting plane method because, through a linearization of the constraints, the feasible region is approximated by a series of hyperplane which cut the box-search space approximating the actual non-linear feasible region boundary. In short, considering

an example problem subjected to only one constraint $g(\boldsymbol{x}) \leq 0$, at the first iteration the starting problem considers only the side constraints,

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u. \end{aligned} \tag{2.3.81}$$

Posing that the optimal solution is $\boldsymbol{x}_1$, at the second iteration the linearization of the constraint is added to the previous problem:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u. \\ & g_2(\boldsymbol{x}) \simeq g_2(\boldsymbol{x}_1) + \nabla g_2(\boldsymbol{x}_1)^T(\boldsymbol{x} - \boldsymbol{x}_1) \leq 0, \end{aligned} \tag{2.3.82}$$

which optimal solution is $\boldsymbol{x}_2$. At the $k$-th iteration the problem will be in the following form

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u. \\ & g_2(\boldsymbol{x}) \simeq g_2(\boldsymbol{x}_1) + \nabla g_2(\boldsymbol{x}_1)^T(\boldsymbol{x} - \boldsymbol{x}_1) \leq 0, \\ & \qquad\qquad \vdots \\ & g_k(\boldsymbol{x}) \simeq g_k(\boldsymbol{x}_{k-1}) + \nabla g_k(\boldsymbol{x}_{k-1})^T(\boldsymbol{x} - \boldsymbol{x}_{k-1}) \leq 0, \end{aligned} \tag{2.3.83}$$

During each iteration a new constraint is added to the previous subproblem providing a new sub-problem because the new linearization have to be centred in the optima of the previous LP sub-problem.
There exist other direct approaches such as the *methods of the feasible directions* which tries to choose the step length $\lambda$ in such a manner that the new point lies in the feasible region. In order to take into account the constraints some different rules were proposed, such as the *Zoutendijk's method* or the *gradient projection* by Rosen, in order to find a feasible direction which also minimize the OF. The *generalized reduced gradient method* (GRG) is based on the elimination of the variables concept. In fact, due to equality constraints (the inequalities are converted in equalities with slack variables), it is theoretically possible to affirm that the design variables are not all independent each other. Therefore, it is possible to split the design vector in two set: one set of the design or independent variables and the other one of the state or dependent variables. In that way, the reduced gradient represent the projection of the $n$-dimensional gradient in a less dimensional feasible space. After compute the reduced gradient, it is possible to proceed as an unconstrained search.
The most popular direct approach is the *sequential quadratic programming method* (SQP), which similarly to the SLP, which is based on a linearization of the OF and the constraints, the SQP is based on a quadratic approximation of the Lagrangian function. Considering a minimization problem with $p$ equality constraints only, the Lagrangian function is

$$L = f(\boldsymbol{x}) + \sum_{k=1}^{p} \lambda_k h_k(\boldsymbol{x}). \tag{2.3.84}$$

Applying the Kuhn-Tucker necessary conditions, they assume the following form

$$\nabla L = \mathbf{0} \text{ or } \nabla f + \sum_{k=1}^{p} \lambda_k \nabla h_k = \mathbf{0} \text{ or } \nabla f + \sum_{k=1}^{p} \boldsymbol{A}^T \lambda_k = \mathbf{0}, \tag{2.3.85}$$

$$h_k(\boldsymbol{x}) = 0, \quad k = 1, ..., p, \tag{2.3.86}$$

where each column of the $n \times p$ matrix $\boldsymbol{A}$ represent the gradient of $h_k$. The above $n+p$ equations in $n+p$ unknowns can be solved adopting the Newton's method [64]. At each $j$-th iteration the value of the increments $\Delta\boldsymbol{x}$ and the $\lambda_{j+1}$ are obtained. It is possible to notice that the final solution at the end of iteration $j$-th is the same to consider the solution of the following QP sub-problem: *Find* $\Delta\boldsymbol{x}$ *such as*

$$\begin{aligned} \min \quad & \nabla f^T \Delta\boldsymbol{x} + \frac{1}{2} \Delta\boldsymbol{x}^T \boldsymbol{H}_L \Delta\boldsymbol{x}, \\ \text{s.t.} \quad & h_k + \nabla h_k^T \Delta\boldsymbol{x} = 0, \quad k = 1, ..., p, \end{aligned} \tag{2.3.87}$$

where $\boldsymbol{H}_L$ is the Hessian of the Lagrangian function (2.3.84). The Lagrangian function of the QP sub-problem (2.3.87) is

$$\tilde{L} = \nabla f^T \Delta\boldsymbol{x} + \frac{1}{2} \Delta\boldsymbol{x}^T \boldsymbol{H}_L \Delta\boldsymbol{x} + \sum_{k=1}^{p} \lambda_k (h_k + \nabla h_k^T \Delta\boldsymbol{x}). \tag{2.3.88}$$

and the Kuhn-Tucker conditions are

$$\nabla f + \boldsymbol{H}_L \Delta\boldsymbol{x} + \nabla h_k \boldsymbol{\lambda} = \mathbf{0}, \tag{2.3.89}$$

$$h_k + \nabla h_k^T \Delta\boldsymbol{x} = 0, \quad k = 1, ..., p. \tag{2.3.90}$$

The original problem can be solved sequentially solving the above QP subproblem if only equality constraints are considered. It is possible to consider also $m$ inequality constraints in the original problem whose Lagrangian become

$$\tilde{L} = f(\boldsymbol{x}) + \sum_{j=1}^{m} \lambda_j g_j(\boldsymbol{x}) + \sum_{k=1}^{p} \lambda_{m+k} h_k(\boldsymbol{x}), \tag{2.3.91}$$

and the QP statement become: *Find* $\Delta\boldsymbol{x}$ *such as*

$$\begin{aligned} \min \quad & \nabla f^T \Delta\boldsymbol{x} + \frac{1}{2} \Delta\boldsymbol{x}^T \boldsymbol{H}_L \Delta\boldsymbol{x}, \\ \text{s.t.} \quad & g_j + \nabla g_j^T \Delta\boldsymbol{x} = 0, \quad j = 1, ..., m, \\ & h_k + \nabla h_k^T \Delta\boldsymbol{x} = 0, \quad k = 1, ..., p. \end{aligned} \tag{2.3.92}$$

The value of the increment at each iteration $\Delta\boldsymbol{x}$ can be considered as a search direction and thus rewrite all the previous QP sub-problems in the design variable $\boldsymbol{s}$. Once the search direction is found, the new position is given by (2.3.60) where the optimal step length can be found minimizing a merit function [59] which is substantially an external penalty function [64]. Once the next position is found, the update of the Hessian can be taken into account with quasi-Newton procedures such as the

BFGS method [59, 64].

The *indirect methods* foresee to solve a sequence of unconstrained problems in order to obtain the solution of the original constrained problem. *Transformation techniques* are suitable only for problems whose constraints are simple explicit functions. In fact, this method tries to perform a change of variable in order to transform the constrained problem in an equivalent unconstrained one. The most used method is a *penalty-function based* approach which will be explained in the following of this Thesis. Another approach is the *augmented Lagrangian multipliers method* (ALM) which combines the Lagrange multiplier method with the penalty function approach. For further readings one can refer to [64].

## 2.4 Meta-heuristic Approaches

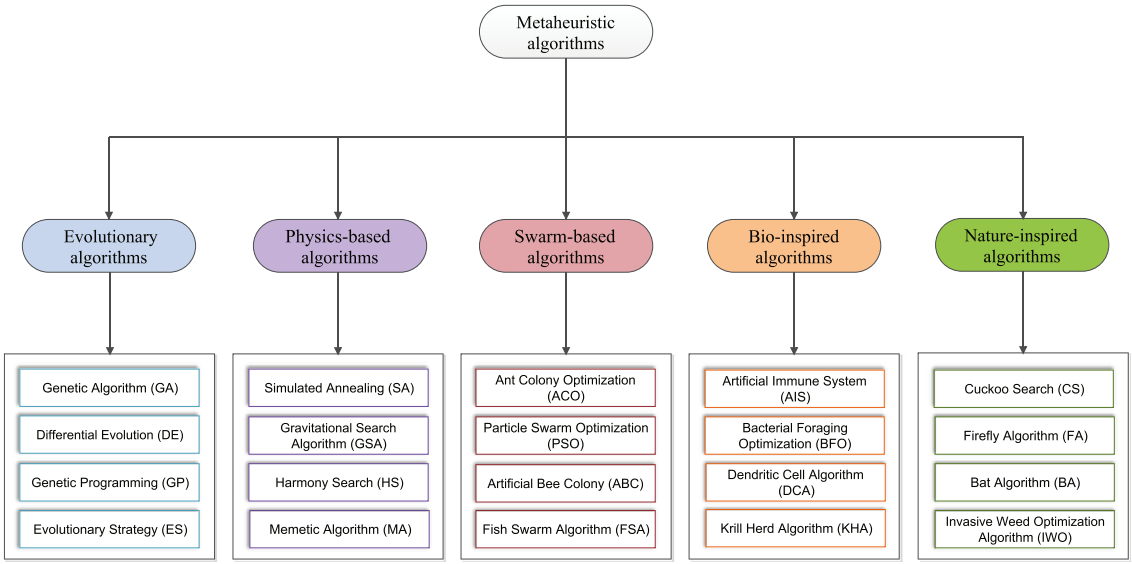



**Figure 2.6:** Image taken from [18]. Classification of meta-heuristic algorithms.

The word heuristic comes from ancient Greek word "*heuretikos*" and it substantially means to discover, find out. In [43], the definition of meta-heuristic by Sörensen and Glover is given:

> "*A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework.*"

From the above definition, it is worth noting that the term indicates a set of concept and strategies developed to find new methods to deal with optimization problems (the high-level framework), but at the same time is related to the specific implementations and algorithms based on this framework which have been proposed during the years starting from the middle of the last century [43].
The meta-heuristic approaches represent nowadays very interesting methods in structural optimization fields, besides the advantage of no prior requirements on differentiability conditions of the OF and constraints. As a matter of fact, as affirmed by [64], engineers are not interested in the exact resolution of the mathematical problem, but they have to find an acceptable design which represents the best solution regarding to some criteria. Therefore, the probabilistic approach given by meta-heuristic algorithms is a very interesting and powerful method to find a "good" solution, respectful of the constraint and quite close to the mathematical exact optimum. This idea is also confirmed in [18], where it is the motivation of a continuous research of new meta-heuristic approaches because of their simplicity in implementation and their effectiveness in solve practical real-life engineering problems. The majority of optimum design engineering problems can be modeled as continuous non-linear optimization problems, in which the original search space is reduced due to the existence

of various constraints. Despite their natural imprecision and not well-posed mathematical theories, the adoption of the meta-heuristic approaches allows to develop new experience-based technique and new methodologies in problem-solving reaching new discoveries in research [43]. Due to this success, they are nowadays adopted to speed up in finding satisfactory engineering results in a reasonable time when classical approaches fail or require too much computational effort. In this part of the present Thesis, firstly a general overview of meta-heuristic algorithms is presented with particular reference to evolutionary. The swarm intelligent *particle swarm optimization* (PSO) algorithm is only mentioned here but it is widely treated in the next chapter.
From a historical point of view, in the 1950s the mathematical stochastic random search methods were completely formalized. They are mathematical optimization methods which are able to deal with problems where the quantities involved are random variables [64]. This allows, for example, to take into account uncertainties in various engineering problems such as in [59]. Then, in the following years, around the 1960s, the meta-heuristic approaches started to develop and they are still under study and research nowadays. The early approaches were also called *artificial intelligence* (AI) but later these two branches take quite different paths notwithstanding they remained to be linked each other [43]. These developments started in those years because of the increase in the availability of more and more powerful computers. The new computational possibilities created the ideal climate for experimenting, testing and demonstrate the effectiveness of meta-heuristic approaches in solving even complex problems in a more efficient way where traditional mathematical methods fail or required too much time or efforts. Despite the human brain have a natural aptitude to the heuristics, the systematic study of the heuristic (for this reason called meta-heuristic) is a relatively new research field [43]. In that sense, it is fundamental to notice that the meta-heuristic approaches were already adopted for such a long time before the coinage of its term around the 1980s. As described in [43], in everyday life the human beings and also animals have to solve optimization problems for their survival and they are quite immediately solved by intuition and past experience, by heuristics, and there is no guarantee that it is the exact optimum. Even in early childhood, the human brain is capable to learn to approximatively solving practical problems without knowing the mathematical programming theory. Some authors argue that it is a legacy from the past, when human beings were still hunter-gatherers and the brain evolves in order to quickly calculate the trajectory of a spear to hit a prey to ensure food resources for their survival in the natural environment. These problems were solved by heuristics because the most important thing was to solve it in a good way as quickly as possible instead of knowing exactly which was the best optimal mathematical trajectory to impose to the spear [43].
The meta-heuristic approaches are not deterministic but they make a strong use of probability concept and, for this reason, are also classified in the so called probability optimization category [59]. As depicted in Figure 2.6 the current available meta-heuristic approaches are many and they can be classified according to their conceptual idea or natural phenomena they try to mimic. As affirmed in [18], regardless of the type of meta-heuristic approach adopted, it is important to focus on the exploration and the exploitation of the algorithm adopted. The exploration means

that the algorithm is able to systematically explore all the areas of the design space in order to find the optimum region. The exploitation means that the algorithm is able to ensure a refined search of the optimum point in a local region. A fine tuning of the balance between these two aspects is really difficult due to the stochastic nature of them, but it will also significantly affect the performance of the algorithm. In fact, an optimal balance may allow to reduce at minimum the computational effort and it may find the optimal solution in few calculus steps regardless of the starting point.

It is always important to keep in mind the *No Free Lunch* (NLF) Theorems (Wolpert and Macready, 1997) for optimization which is pragmatically summarized in [59]:

> "*There is no optimization algorithm capable of handling efficiently every optimization problem.*"

This statement justifies why it is still needed to explore searching for new algorithms and new approaches in order to find the best suitable one for a specific class of problems, but, in particular, making comparisons with other techniques and do not only rely on one single meta-heuristic approach.

### 2.4.1 Evolutionary algorithms

Evolutionary algorithms are the first class of meta-heuristic which have been implemented and they take inspiration by what naturally happens in the biological context where a population of individuals evolves, according to the Darwinian Theory of Evolution and natural selection, to better adapt to the existing environment [59]. This paradigm can be ideally transposed to a numerical procedure which leads a population of trial solutions to evolve toward the global optimum of the optimization problem adapting to the environment which is represented by the feasible region reduced by the presence of the constraints. Substantially, these agents are competing for the resource and only the survival of the fittest to the environment will pass to the next generation [62]. Evolutionary algorithms (EAs) can be considered general and versatile tools for solving constrained optimization problems. The research interest for this class of optimizers is continuously increasing, mainly because objective functions (OF) and constraints are not required to be differentiable, continuous, or even explicit. In addition, no preliminary assumptions or a priori information are needed for solving constrained optimization problems by means of EAs. Moreover, EAs have a better global search ability compared to traditional numerical strategies (i.e., gradient-based algorithms) and a good starting design is not essential, since they operate on a population of individuals (coded candidate solutions of the optimization problem) which are efficiently handled during the evolutionary search. In contrast, EAs lack of well-posed theories about their convergence and a large number of function evaluations and computational effort may be needed to converge. Moreover, Evolutionary optimization algorithms were originally developed for solving unconstrained optimization problems, thus, they naturally lacked of a mechanisms to handle the constraints of the problem at hand. Nonetheless, the

unavoidable existence of several restrictions either from a mathematical and an engineering point-of-view leads to a huge contraction of the available design space, thus, reducing the number of the admissible solutions. It is obvious that the resolution of constrained optimization problems is much more complicated, especially when a large number of constraints are involved which can reduce the size and increase the complexity of the feasible domain dramatically. In these circumstances, both effectiveness and correctness of the evolutionary-based search can be jeopardized and the final results can be unsatisfactory. In fact, it is very probable to achieve an infeasible final solution if the search of the best objective function value does not efficiently take into account the imposed constraints. In contrast, the optimizer could be entrapped into a sub-optimal area if the exploration of the search space is conducted by taking into account feasible solutions only. Therefore, the implementation of effective constraint-handling mechanisms is considered a crucial issue for all biological inspired optimizers, [17, 10, 81, 45]. It is much evident that a competitive technique for handling constraints in evolutionary computation should be able to achieve the best possible compromise between conflicting requirements. In the following, some of the most important EAs are mentioned.

The first formal systematic study of the *Genetic Algorithm* (GA) is related to the work of John Holland in 1975 at the University of Michigan, and it was extended later mainly by his student David E. Goldberg [43]. In reality, the first ideas about biological evolutions started around the 1960s with the mathematician Barricelli at first and the biologist Rechenberg after which are considered the fathers of the artificial life research [64]. The GA takes inspiration from the genetic field. The characteristics of an individual are transmitted to the offspring through chromosome which contains a series of information called alleles. The set of alleles is called genotype. These genetic information are traduced in the external observable world as physical characteristics which are called the phenotype. From this paradigm, the GA implement numerical operators which try to mimic this mechanism. In this approach, a population of $n$ individuals composed by a set of design vectors, also called trial vectors, is considered. Each individual is encoded in binary strings forming a chromosome in which each bit is an allele [59]. When no further information is available, the population is initialized randomly and to preserve the diversity the Latin hypercube sampling is suggested [43]. The dimension of the population is suggested to be at least from $2n$ to $4n$ if $n$ is the number of the design variables [64]. As reported in [59], Goldberg proposed to set the population size according to this relation $n_{pop} = 2^k(q/n)$ where $q$ is the length of the binary string. If each design variable is coded as a $q$ length string, a single trial vector will be a string whose total length is $nq$. Due to this representation, the GA naturally started as an optimization procedure for discrete or integer problems. In fact, given a binary string in the form $b_q b_{q-1} b_{q-2} ... b_1 b_0$, the equivalent decimal number is $y = \sum_{k=0}^{q} 2^k b_k$. This discrete representation can be also used for a continuous variable with side constraints ($x_i^{(l)} \leq x_i \leq x_i^{(u)}$) using this conversion formula

$$x_i = x_i^{(l)} + \frac{x_i^{(u)} - x_i^{(l)}}{2^q - 1} \sum_{k=0}^{q} 2^k b_k, \tag{2.4.1}$$

but, in that case, to obtain a sufficient accuracy $\Delta x$, it is necessary to adopt a large number of bit $q$ according to this relation

$$2^q \geq \frac{x_i^{(u)} - x_i^{(l)}}{\Delta x} + 1. \tag{2.4.2}$$

There are also possible different representations such as a variable-length array or the random key encoding when the permutation and sorting order have to be taken into account such as in the Traveling Salesman problem [43]. Nowadays is also possible an implementation with a real encoding representation of the design variable.
The GA is based on the Darwin's theory of survival of the fittest. In practice by a crude imitation of what happens in Nature, species whose individuals are better evolve to the next generation. In numerical terms, the trial solutions in the population will compete each other because only the individuals with a highest fitness (the phenotype in traditional genetics [43]) will survive to the next generations. In this terms, the fitness evaluation $F(\boldsymbol{x})$ is substantially the OF evaluation $F(\boldsymbol{x}) = f(\boldsymbol{x})$ and, for this reason, the GA started as a resolution method for a maximization problems [64]. GA started as an unconstrained method, therefore it is possible to adopt indirect methods such as the penalty functions to transform the constrained problem in an unconstrained one. A minimization problem can be easily converted in a maximization one, but in terms of GA this conversion may be more useful if the fitness function is non-negative [64]

$$F(\boldsymbol{x}) = \frac{1}{1 + f(\boldsymbol{x})}. \tag{2.4.3}$$

The core of the GA is based on three mathematical operators which occur in this order at each generation: reproduction, crossover and mutation. At the final step, the convergence criteria are checked and if they are not still reached, the GA cycle starts again.
The *reproduction operator* can be seen as a selection operator because randomly select which elements will enter in the mating pool to generate the next generation. The probabilistic procedure of the selection procedure is based on the OF of each individual associating to each string a selection probability $p_i$ to enter in the mating pool,

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j}, \quad i = 1, ..., n. \tag{2.4.4}$$

The concept can be transposed to a spinning roulette wheel with a selection pointer. Its circumference is divided into segments and each segment length is proportionally to the extraction probability. In this way, the best individuals will have a greater probability to be selected than the worse individuals. It is also possible to assume a selection rule based on the cumulative probability of $i$-th string $P_i = \sum_{j=1}^{i} p_j$. The roulette wheel circumference is divided into segments whose length is proportional to $(P_{i-1} - P_i)$. The selection of an individual is thus performed generating a random number between 0 and 1 and picking up the string corresponding to this number. As it is possible to notice the fittest feasible members are encouraged to be selected but the worst and maybe infeasible members are not excluded at all because they

contain important information when the optimum is located at the boundary of the feasible region [62]. As underlined in [64], in this phase, no new string is generated but they are only selected from the original population to create the mating pool. Other many selection rules were also be developed in years. One of the most popular methods is the stochastic universal selection of Baker [43] in which the roulette wheel presents a number of spinners (selection pointer) equally spaced around the circumference. This procedure allows to reduce the stochastic probability to extract some individuals too many times or not enough times. Another possible approach is the Tournament selection operator which randomly extracts different small sets of individuals from the population. As exposed in [62], the fittest individual of each set will enter into the mating pool. The size of the subsets is called tournament size and it is typically varying between 2 (binary tournament [43]) and 4 . As stated in [59], this latter approach appeared to Goldberg to be more efficient than the classic roulette wheel rule. The ranking scheme sorts the individuals from the best to the worst according to their fitness but then assigns to each chromosome a selection probability based in their rank and not directly on the value of the OF. In the elitist approach, in order to avoid losses of information regarding to the best current individual, the fittest member or a small set of the fittest individuals directly pass to the next generation without any reproduction stage [59, 62].
To the *crossover operator*, also called recombination operator in [62], is now demanded to generate new strings starting from two parents randomly chosen in the mating pool. The new offspring will be a combination of information contained in the parents. To perform this combination several rule of crossover were exposed in the years. The most simple rule is the so called single-point crossover which randomly defines a crossover point among the alleles the parents and swaps one of the resulting sub-strings between the two parents forming two new offspring strings [64]. For instance, from two parent strings

Parent 1: 10010|1010111
Parent 2: 01011|1000001

two child strings can be obtained using the single-point crossover

Offspring 1: 10010|1000001
Offspring 2: 01011|1010111

There exist also a multi-point or $n$-point crossover which randomly determine $n$ crossover points. As stated in [59], the uniform crossover is a generalization of the single-point crossover. In this strategy, a random binary string of the same length of the parents' chromosomes is generated and it is logically decoded as a crossover mask. If a bit of the mask is 1 the alleles associated with the parents corresponding to the mask's bit position are swapped between the two parents, otherwise if the bit of the mask is 0 they are not swapped. This operator performs in a better way because it enables to increase the diversity between the population members. Below

an example of uniform crossover is presented:

| | |
|---|---|
| Parent 1: | 100101010111 |
| Parent 2: | 010111000001 |
| Mask: | 100011000001 |
| | ↓ ↓↓ ↓ |
| Offspring 1: | 000111010111 |
| Offspring 2: | 110101000001 |

Due to its random nature, the resulting offspring may or not be better than the parents. In that latter case, no great issues are arising because the selection process will discard the worst in the next generation. In reality, to preserve the integrity of some good existing parents and avoiding losses of information, a crossover probability $p_c$ is also set. In that way, only $100p_c$ percent of mating pool's strings will be subjected to crossover and the others $100(1 - p_c)$ percent remains unchanged.
If some good information are lost, the selection and the crossover operators may not be able to recover them. In that sense, a further random operator, the *mutation operator* can perform a random perturbation of a bit change in one allele. This is done mainly to allow al locally alteration of an allele in order to recover lost alleles during the various generations [59]. Furthermore, the mutation operator allows to take into account the random changing which leads to the appearance of new possible unseen characteristics which occur in a real population. This helps to avoid premature convergence because it enables to maintain the diversity among the individuals [59].

Original String: 010111010111

Mutated String: 010101010111

The mutation is a rare event also in real populations, therefore it is taken into account with a mutation probability $p_m$ priorly fixed. In the GA, a random number is generated between 0 and 1 and if this number is greater than $p_m$ the mutation does not occur, otherwise it does.
The convergence criteria can be set in two way [64]: the first is trivially considering a maximum number of allowable generations $k_{max}$; the second criterion can be set when the fitness of the entire population reach a standard deviation value $\sigma_{max}$ lower than a certain small positive constant.
Quite early, many variants of the standard GA were also proposed; for instance, the real-encoded GA become popular for its simpler implementation due to the fact that the binary encoding process was not adopted [62]. The principles of the various GA operators are always the same, but they have to be adapted to operate with real numbers and no more with binary strings. For instance, in [62] the heuristic crossover operator generate new offspring ${}^{k+1}\boldsymbol{o}_1$ for the next generation selecting two random parents from the mating pool ${}^{k}\boldsymbol{x}_1, {}^{k}\boldsymbol{x}_2$ following the below relation,

$$ {}^{k+1}\boldsymbol{o}_1 = {}^{k+1}r \left( {}^{k}\boldsymbol{x}_1 - {}^{k}\boldsymbol{x}_2 \right) + {}^{k}\boldsymbol{x}_1, \tag{2.4.5} $$

where ${}^{k+1}r = rand[0, 1]$ is a random number between 0 and 1 and the parent ${}^{k}\boldsymbol{x}_1$ is assumed to be the fittest ($f({}^{k}\boldsymbol{x}_1) < f({}^{k}\boldsymbol{x}_2)$). Other real-valued crossover operators are also presented in [62] such as the Laplace crossover and the average-bound

crossover. The same adaptation need to be performed to the mutation operator for instance adopting the wavelet mutation presented in [62] by Ling and Leung. In any case if the offspring after crossover and mutation lie in the unfeasible region, it can also be reinitialized randomly.
Due to the GA random nature, unfortunately there are no mathematical proofs of its convergence, but numerically studies demonstrated than they are able to succeed also dealing with highly non-linear, non-convex and discontinuous domains. Furthermore, since the GA require only the OF evaluation and not require any information about the gradient, its computational effort will be lower compared with mathematical programming techniques such as NLP methods [59]. Furthermore, in literature, it is possible to see many proposals to improve the convergence of GA, to dynamically adapt the various operators in order to promote exploration in the first generations and enhance the exploitation towards the end [62]. Also the prior user-defined values of the hyper-parameters, such as the population size, the probability of crossover and the probability mutation, has to be carefully chosen but no general guideline are available but only literature suggestions and a fine tuning trial-and-error search is always necessary [62].

The *Evolution Strategies* methods (ES) were formalized by Rechenberg and Schwefel at the Technical University of Berlin around the 1960s [43]. They are based on the biological imitation similar to GA but, in particular, referring to the branch called organic evolution. The ES was early adopted also in structural optimization problems with dynamic load and earthquake design [59]. There exist two main categories of ES. The first one is related to the so called *two-membered ES* in which only one parent is able to produce only one offspring. The second category is the *multi-membered ES* in which a population of $n$ parents can produce in general $m$ offspring.
The two-membered ES, also called (1+1)-ES in [2], operates in two steps. In the first step a mutation operator, different than GA, occurs and generate an intermediate population. At the generation $k$, the new offspring $\boldsymbol{x}_{o,k}$ is generated by the parent $\boldsymbol{x}_{p,k}$ adding a random $n$-tuple $\boldsymbol{z} = [z_1, ..., z_n]^T$:

$$\boldsymbol{x}_{o,k} = \boldsymbol{x}_{p,k} + \boldsymbol{z}. \tag{2.4.6}$$

The name of this operator is due to the fact that the mutation is an event which rarely occurs with great changes and more frequently provides very little changes. This idea lead to choose $\boldsymbol{z}$ from a normal or Gaussian probability distribution $\mathbf{N}(0, \sigma_i)$ with nil mean and a standard deviation $\sigma_i$ [59]. To choose an appropriate standard deviation, also called mutation strength in [2], some tuning approaches can be done because $\sigma_i$ directly affect the rate of change. This is because it represents a sort of average value of the length of the random step length to which the parents are subjected to generate the offspring. If it is chosen too large, the optimum point could be easily straddled, whereas if it is too low the convergence will be too slow. It is worth noting that, using a Gaussian bell-shaped distribution, the algorithm is mainly promoting the exploration in a certain neighbourhood of the selected parent. In the second step a selection operator define which individual will survive and become the parent at the next generation $x_{p,k+1}$. The selection operate in a binary

competition between the offspring and the parent

$$x_{p,k+1} = \begin{cases} \boldsymbol{x}_{o,k} & \text{if } \boldsymbol{x}_{o,k} \text{ is feasible and } f(\boldsymbol{x}_{o,k}) \leq f(\boldsymbol{x}_{p,k}) \\ \boldsymbol{x}_{p,k} & \text{otherwise} \end{cases} \tag{2.4.7}$$

In an advanced later version of the two-membered ES, the multi-membered ES started to implement also a recombination operator [43]. To increase the exploration the so called $(\mu + \lambda)$-ES scheme was adopted. Substantially, a set with $\mu$ parents is selected from the $n$ members of the current population. This set of parents is able to produce $\lambda \geq 1$ offspring. From this set of parents and offspring, a $\mu + \lambda$ intermediate population is subjected to a deterministic selection operator able to produce the next generation. In order to maintain the population size $n$, $\lambda$ members are discarded from the intermediate population of $\mu + \lambda$ individuals based on their fitness according to a natural selection scheme [2, 59]. Another variant of the multi-membered ES is the so called $(\mu, \lambda)$-ES scheme where the selection operator acts only on the $\lambda$ offspring and the parents are lost regardless if they were good or not. in this way, to maintain the population size, the recombination operator have to generate $\lambda > \mu$ offspring [2]. Since the parents are not considered in the selection phase, each individual can not live more than one generation [59]. This latter scheme were numerically proven that provided good results in dynamic problem (where the optimum change in time) and with noisy OF [59].
Regardless the multi-membered ES selection scheme, the recombination operator is quite different from GA because it can produce offspring also from one, two or even more parents. As reported in [43], the intermediate recombination is able to produce offspring from an average of a set of parents, whereas the Weighted multi-recombination operates in a similar way but adopting a weighted mean based on the fitness of the parents. In [59] is presented a recombination rule in which, for $n$-dimensional continuous problems, the $\mu$ parents' population is composed by a set of temporary $n$-dimensional parent vectors $\tilde{\boldsymbol{x}} = [\tilde{x}_1, ...\tilde{x}_n]^T$. Each $i$-th component $\tilde{x}_i$ of each temporary parent vector can be chosen adopting one of the following strategies

$$\tilde{x}_i = \begin{cases} x_{i,a} \text{ or } x_{i,b} \text{ randomly} & (a) \\ 0.5 \cdot (x_{i,a} + x_{i,b}) & (b) \\ x_i^{rand} & (c) \\ x_{i,a} \text{ or } x_i^{rand} & (d) \\ 0.5 \cdot (x_{i,a} + x_i^{rand}) & (e) \end{cases} \tag{2.4.8}$$

where $x_{i,a}$ and $x_{i,b}$ are the $i$-th components of two random vector chosen from the current population $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, whereas the term $x_i^{rand}$ is the $i$-th component of a randomly chosen temporary parent vector already defined. In [59], two convergence criteria are proposed: the first check if the absolute or relative difference of the OF between the worst and the best individual is less than a small positive constant $\varepsilon_1$; the second criterion check if the mean of the OF has stalled to the same value, within a relative tolerance $\varepsilon_2$, for at least the last $2n$ generations.
In modern approaches, some new principles as the strategy parameters (which control the probability distribution for each design variable), the self-adaptation concept

or some hybridization with some mathematical methods (such as in the is covariance matrix adaptation CMA-ES) were also implemented to make more robust and reliable search processes in ES [43].

The *Differential Evolutionary* algorithm (DEa or simply DE) is one of the most recent stochastic, population-based global optimization method which stay under the umbrella of the EAs which was proposed by Storn and Price in the 1997 [23, 62]. It is always based on the natural evolution process but it operates in a quite different way. The main difference with GA is not only the framework of the selection, mutation and crossover operators but also the order they act. In fact, in the DE, such as in ES, the mutation operator come first of the crossover operator and, in this approach, it represents the explorative operator. As usual in EAs, at first, a population of $N$ design vectors with $n$ design variables is usually randomly initialized adopting the Latin Hypercube Sampling (LHS) in order to generate an initial population with the minimum correlation between samples [23]. After that, the mutation operator randomly choose two vectors from the population and calculate the differences between them term by term. This geometric approach is what denote the name of the procedure and it means that the DE tries to estimate the gradient in that zone rather than locally in a single point. This difference, weighted in some way by the mutation operator, is added to a third randomly chosen vector in the population which is called *target vector*. The resulting vector obtained from the mutation operator is called *mutant vector*. The mutation operator act in order to generate a mutant vector ${}^{k+1}\boldsymbol{v}_i$ for the iteration $k+1$ for each member of the original population at the iteration $k$. There are also many variants in literature which consider different weights or different number of vectors needed in the difference calculus. These frameworks are usually denoted as *DE/a/b/c* in which the $a$ represent the way to construct the mutant vector, $b$ is the number of random differences considered and $c$ is referred to a specific type of crossover operator. As reported in [23, 62], the most used framework are the following

$$\textit{rand/1/bin} \qquad {}^{k+1}\boldsymbol{v}_i = {}^{k}\boldsymbol{x}_{r_1} + F_1({}^{k}\boldsymbol{x}_{r_2} - {}^{k}\boldsymbol{x}_{r_3}), \tag{2.4.9}$$

$$\textit{best/1/bin} \qquad {}^{k+1}\boldsymbol{v}_i = {}^{k}\boldsymbol{x}_{best} + F_1({}^{k}\boldsymbol{x}_{r_1} - {}^{k}\boldsymbol{x}_{r_2}), \tag{2.4.10}$$

$$\textit{current-to-best/1/bin} \qquad {}^{k+1}\boldsymbol{v}_i = {}^{k}\boldsymbol{x}_i + F_2({}^{k}\boldsymbol{x}_{best} - {}^{k}\boldsymbol{x}_i) + F_1({}^{k}\boldsymbol{x}_{r_1} - {}^{k}\boldsymbol{x}_{r_2}), \tag{2.4.11}$$

$$\textit{best/2/bin} \qquad {}^{k+1}\boldsymbol{v}_i = {}^{k}\boldsymbol{x}_{best} + F_2({}^{k}\boldsymbol{x}_{r_1} - {}^{k}\boldsymbol{x}_{r_2}) + F_1({}^{k}\boldsymbol{x}_{r_3} - {}^{k}\boldsymbol{x}_{r_4}), \tag{2.4.12}$$

$$\textit{rand/2/bin} \qquad {}^{k+1}\boldsymbol{v}_i = {}^{k}\boldsymbol{x}_{r_1} + F_2({}^{k}\boldsymbol{x}_{r_2} - {}^{k}\boldsymbol{x}_{r_3}) + F_1({}^{k}\boldsymbol{x}_{r_4} - {}^{k}\boldsymbol{x}_{r_5}), \tag{2.4.13}$$

where $r_1, r_2, r_3, r_4$ are random integer value chosen in the set $\{1, ..., i-1, i+1, ..., N\}$, ${}^{k}\boldsymbol{x}_{best}$ denote the fittest individual at the iteration $k$ and $F_1, F_2$, called mutation coefficients, are real positive constants which weight and scale the differences and govern their amplitude. There are some literature suggestions which say to adopt a constant value of 0.5 for the mutation coefficients. The adoption of this type of mutation permit to enhance the diversity between the population members [23]. As stated in [62], a projection scheme is usually adopted after mutation in order to verify the satisfaction of the side constraints ($x_j^{(l)} \leq x_j \leq x_j^{(u)}$) for each $j$-th design

variable. One possible simple approach is

$$ {}^{k+1}v_{ij} = \begin{cases} x_j^{(l)}, & \text{if } {}^{k+1}v_{ij} < x_j^{(l)}, \\ x_j^{(u)}, & \text{if } {}^{k+1}v_{ij} > x_j^{(u)}, \\ {}^{k+1}v_{ij}, & \text{otherwise.} \end{cases} \tag{2.4.14} $$

After the mutation operator, the mutated vector ${}^{k+1}\boldsymbol{v}_i$ and the corresponding original vector ${}^{k+1}\boldsymbol{x}_i$ are subjected to the crossover operator. The two main used crossover operators in DE are called exponential crossover (labelled as $DE/a/b/exp$) and binomial crossover (labelled as $DE/a/b/bin$). As affirmed in [43], they respectively remind the two-point crossover and the uniform crossover of the GA. As stated in [23], the binomial crossover generates an offspring ${}^{k+1}\boldsymbol{u}_i$, called *trial vector*, adopting the following rule for each $j$-th component of the $i$-th trial vector

$$ {}^{k+1}u_{ij} = \begin{cases} {}^{k+1}v_{ij}, & \text{if } u \leq p_c \text{ or } j = randint[1, n], \\ x_{ij}, & \text{otherwise,} \end{cases} \tag{2.4.15} $$

where $u$ is a pseudo-random number drawn in the interval $[0, 1]$ with a uniform probability distribution whereas $p_c$ is the crossover probability (also called crossover ratio or probability of reproduction) priorly defined by the user. There are some literature suggestions for a proper choice of $p_c$, e.g. Storn and Price suggest to adopt a constant value between 0.8 and 1. The index $j$ is a integer randomly chosen in the set $\{1, ..., n\}$, with $n$ total number of the design variables. This latter additional condition is adopted in order to ensure that the offspring vector and the original vector differs for at least one component [62]. In practice, this ensures that at least one term of the mutated vector is taken into account in the construction of the offspring trial vector.
After that, if the optimization problem is unconstrained, the selection operator simply acts as a one-to-one competition, based on the OF evaluation, between the offspring ${}^{k+1}\boldsymbol{u}_i$ and the original vector ${}^{k}\boldsymbol{x}_i$ in order to define the new design vector ${}^{k+1}\boldsymbol{x}_i$ at the next $k+1$ generation,

$$ {}^{k+1}\boldsymbol{x}_i = \begin{cases} {}^{k+1}\boldsymbol{u}_i, & \text{if } f({}^{k+1}\boldsymbol{u}_i) < f({}^{k}\boldsymbol{x}_i), \\ {}^{k}\boldsymbol{x}_i, & \text{otherwise.} \end{cases} \tag{2.4.16} $$

If the DE is dealing with a constrained problem, the selection operator is modified in order to naturally implement a strategy to handle the constraints adopting the Deb feasibility rules and the dominance concept. This latter has been already explained in the Section 2.2 of the present Thesis. As reported in [23], considering the original design vector ${}^{k+1}\boldsymbol{x}_i$ and its offspring ${}^{k+1}\boldsymbol{u}_i$, it is possible to say that ${}^{k+1}\boldsymbol{u}_i$ is dominated by ${}^{k+1}\boldsymbol{x}_i$ and this condition is written in symbols as ${}^{k+1}\boldsymbol{x}_i \prec {}^{k+1}\boldsymbol{u}_i$. In order to consider the feasibility, the Deb's rule permit to preferably choose the feasible members respect to the unfeasible ones taking into account the degree of violation of the constraints. This violation can be written for the $i$-th member as

$$ \Phi({}^{k+1}\boldsymbol{x}_i) = \sum_{p=1}^{n_p} \max\{0, g_p({}^{k+1}\boldsymbol{x}_i)\} \geq 0, \tag{2.4.17} $$

which is nil if and only if all the constraints are all satisfied. As stated in [23], the feasibility dominance selection rule can assume the following form

$$ {}^{k+1}\boldsymbol{x}_i \prec {}^{k+1}\boldsymbol{u}_i \Leftrightarrow \begin{cases} f({}^{k+1}\boldsymbol{x}_i) < f({}^{k+1}\boldsymbol{u}_i) \text{ and } \left(\Phi({}^{k+1}\boldsymbol{x}_i) = 0\right) \wedge \left(\Phi({}^{k+1}\boldsymbol{u}_i) = 0\right), \\ \left(\Phi({}^{k+1}\boldsymbol{x}_i) = 0\right) \wedge \left(\Phi({}^{k+1}\boldsymbol{u}_i) > 0\right), \\ \Phi({}^{k+1}\boldsymbol{x}_i) < \Phi({}^{k+1}\boldsymbol{u}_i), \quad \text{with } \Phi({}^{k+1}\boldsymbol{x}_i), \Phi({}^{k+1}\boldsymbol{u}_i) > 0. \end{cases} \tag{2.4.18} $$

As explained in [23], the previous conditions say that ${}^{k+1}\boldsymbol{x}_i$ dominate ${}^{k+1}\boldsymbol{u}_i$, and therefore ${}^{k+1}\boldsymbol{x}_i$ will survive in the next generation when: 1) both are feasible but ${}^{k+1}\boldsymbol{x}_i$ has a lower OF than ${}^{k+1}\boldsymbol{u}_i$; 2) ${}^{k+1}\boldsymbol{x}_i$ is feasible and ${}^{k+1}\boldsymbol{u}_i$ not; 3) both are unfeasible but ${}^{k+1}\boldsymbol{x}_i$ is preferable because it has the minimum degree of violation. This static dominance scheme ensure the survival of feasible solution respect to the unfeasible ones.
Finally, as usual, a convergence stopping criterion needs to be adopted. Usually, a maximum number of iterations is set, despite it is not known a priori which is the best value to adopt but it is always necessary to perform some trial-and-error or adopting some literature suggestions. As stated in [43], the DE has a natural self-adapting behaviour because during the process it is able to automatically modify the direction and the entity of the movements because the differences tend to decrease if the population converge toward the optima. As reminded by [62], the main issue in adopting DE is the proper choice of their hyper-parameters starting from the population size. As a matter of fact, a too small population will lead to premature convergence, whilst a too large size will lead to stagnation and a high computational effort. In literature, it is suggested to adopt a size in the order of magnitude of ten times the number of the design variables to optimize. Many other variants of the standard DE were proposed during the years to mitigate these drawbacks e.g. introducing some hybridization or, for instance, some adaptive control parameters in which there is a dynamically choice of the mutation coefficients or the probability of crossover.

### 2.4.2 Physics-based, Bio-inspired, Nature-inspired Algorithms

The *physics-based algorithms* take inspiration by intelligent agents which explore the design space and whose movements are governed by some physics rules such as gravitational, electromagnetic or inertia force and many others [18]. One of the most popular algorithms is the *Simulated Annealing* (SA) initially developed in 1983 by Kirpatrick [82] which has its roots in the cooling process of metals in metallurgy processes [64]. When a metal is heated beyond its fusion critical temperature, it reaches its molten state. In this physics state of matter, the particles are free to move because of its fluid nature and the high kinetic energy of the atoms due to the high thermal energy level. As the temperature starts to decrease, particles tend to lose their kinetic energy and gradually find a position in the crystalline lattice of the material with the minimum internal energy until it again comes back to the solid state. In metallurgy, if the cooling process is fast, the particles are not able to reach

a compact state and the polycrystalline state of the metal will be characterized by a disorganized structure with larger grains and both significant internal stresses in the material. If the cooling process is slow the atoms have enough time to reach the ideal position with minimum internal energy and the resulting polycrystalline structure will have much ordered structure and fine grains and quasi-nil internal stresses. This latter process used in metallurgy and both in glass production [43] is known as annealing [64]. The SA tries to simulate this process taking into account of a temperature parameter $T$ and adopting the Boltzmann's probability distribution. Posing that $k$ is the Boltzmann's constant, this law explains that the energy $E$ of a body in thermal equilibrium at a temperature $T$ is distributed according to the following distribution law

$$P(E) = e^{-\frac{E}{kT}}. \tag{2.4.19}$$

The term $P(E)$ represents the probability of achieving the level of energy $E$ at that fixed temperature. From this negative exponential rule, it is possible to notice that at high temperatures the probability is quite uniform for each energy state level, whereas at low temperatures the system has a low probability to have a high energy level. As stated in [64], Metropolis et al. around 1950s try to adopt this principle in the minimization problems associating the OF evaluation in $\boldsymbol{x}_i$ to the energy state of a thermodynamic system $E_i$,

$$E_i = f_i = f(\boldsymbol{x}). \tag{2.4.20}$$

The Metropolis criterion establishes that the probability of the next design point $\boldsymbol{x}_{i+1}$ chosen randomly in the design space is related to the difference of the energy respect to a design reference state $\boldsymbol{x}_i$ at a fixed temperature level$T$,

$$\Delta E = E_{i+1} - E_i = \Delta f = f_{i+1} - f_i, \tag{2.4.21}$$

and, adopting the (2.4.19), the probability associated to this new state $\boldsymbol{x}_{i+1}$ is

$$P(E_{i+1}) = \min\left\{ 1\,;\, e^{-\frac{\Delta E}{kT}} \right\}. \tag{2.4.22}$$

Despite its physical meaning, the context of the SA the Boltzmann's constant $k$ acts merely as a scaling factor, therefore it can be set to a unity value. When the result of (2.4.21) is negative, the new state $\boldsymbol{x}_{i+1}$ is accepted with a unitary probability because it give a lower value of the OF. On the contrary, if this value is positive, it means that the new point is related to a worse value of the OF and, in a normal optimization process, it would simply be discarded. In the SA instead it is accepted with a certain probability, it means that, generating a random number $z$ between 0 and 1, if this random number is greater than the probability value obtained by (2.4.22) the new point $\boldsymbol{x}_{i+1}$ is accepted, otherwise it is discarded. In symbols

$$\begin{cases} \text{if } z = rand[0,1] \geq P(E_{i+1}) = e^{-\frac{\Delta E}{kT}} < 1 \Rightarrow \boldsymbol{x}_{i+1} \text{ is accepted} \\ \text{otherwise } \boldsymbol{x}_{i+1} \text{ is discarded} \end{cases} \tag{2.4.23}$$

This process is one iteration and at a fixed level of temperature $T$ the total number of iteration is $n$. In literature, it is suggested to choose $n$ in an interval from 50 to

100 but there is not a unique choice for this hyper-parameter and a trial and error search need to be performed for the specific problem under study. In fact, it can strongly affect the convergence of the procedure: if $n$ is too large, the computational effort will be high but the exploration level will be great, whereas if $n$ is too small, it may not explore the entire design space resulting in a premature convergence to a local optimum. After performing $n$ iteration at a fixed temperature level $T$, the temperature is reduced to the next level by a rational reduction factor $c$ called cooling schedule, with $0 < c < 1$, which is directly multiplied to the temperature $cT$. In literature, it is suggested to choose $c$ between 0.4 and 0.6. Also here this hyper-parameter strongly affect the performance of the SA, because an excessively small value will lead to a strong reduction of $T$ and thus the probability which not allow a good exploration of the design space, whereas a high value of $c$ will lead to an extremely high computational effort because too many steps will be required to perform an imposed interval of thermal variation. The possible convergence criteria are related to reaching a relatively small value of temperature or when the $\Delta f$ becomes relatively small. Similarly to EAs, also in the SA, there are no requirements on the differentiability of the OF and the constraints. For problems involving constraints satisfaction, an indirect method with penalty function to transform the problem in an unconstrained problem is also possible [64].

In the group of the physics-based algorithm, there are also some other techniques which are inspired by human behaviour. Some of the most popular examples of such techniques are the *Harmony Search* (HS) and and the *Tabu Search* (TS) [18].

According to the classification in the Figure 2.6, the *Bio-inspired* algorithms take inspiration from the biology field and adopt their models to solve engineering problems. For instance, one of the most popular techniques is the *Artificial Immune System* (AIS). It tries to use the paradigm of the adaptive molecular processes of the human immune system which is able to detect, recognize and tackle the pathogen substances and dangerous for the human body [43]. As a matter of fact, it can be considered as an intelligent system because it is also able to learn from the past, therefore it possesses a memory. The immune system is a complex system composed of tissues, organs, cells and specific agents such as the antibodies which collaborate together in a very coordinate way. Furthermore, the AIS is able to solve complex classification problems and recognize the various external biological agents, harmful substances, bacteria, viruses which the human body comes in contact with. As stated in [53], the AIS has its roots in the theoretical immunology and the empirical observed immune functions. All of this aspect from the biological field were transposed to the problem-solving techniques and soft computing and computational intelligence. This metaphor inspired researchers to start to develop this system from the 1990s, reaching nowadays some robust methods which are applied in multi-objective optimization problems and for pattern recognition problems [43]. For instance, In [53], the AIS was used to solve multi-objective structural optimization problems of composite structure to find the best design with minimum weight and cost. The framework of the AIS foresees the adoption of a mutation operator, a cloning operator (which is an asexual reproduction operator) and some memory cells which are used to store the best performing search agents during the evaluation and the

selection phase. A sort of evolution is also used in the AIS because the agents can die in a final stage and new better search agents are generated by cloning. Some specific strategies are also adopted to maintain diversity in the population [53].

Despite the majority of the meta-heuristic algorithms take inspiration by Nature, as presented in the Figure 2.6, in [18] the *Nature-inspired* algorithms are treated as an apart class which collects all the methods which are not included in the other classes. In particular, these methodologies take inspiration by specific aspects of natural behaviours of certain animals. To cite an example, the Cuckoo Search (CS) algorithm developed by Xin-She Yang and Suash Deb in 2009 mimics the reproduction behaviour of the cuckoos which is characterized by the so called brood parasite [82]. This algorithm was also adopted to solve structural optimization problems, one can refer for example to [24].
In the 2004 and 2005 Artificial Bee Colonies (ABC) algorithm was developed. It takes inspiration by the hierarchical society of honey-bees which congregates in colonies and in each of them there are three different types of bees: the worker bees, the queens and the drones. The workers bees conduct the most important tasks for foraging because they collect and store the honey. Biologists observed that bees can communicate each other using the so called "waggle dance", therefore they perform some social interaction [82]. In the ABC, a prior user-defined number $N_s$ of "food sources" (local attractors) is set in the design space adopting the Monte Carlo randomization respecting the side constraints ($x_j^{(l)} \leq x_j \leq x_j^{(u)}$ with $j = 1, ..., n$ where $n$ in the number of the design variables) of each design variables

$$x_{i,j} = x_j^l + rand[0,1] \cdot (x_j^u - x_j^l), \quad i = 1, ..., N_s \tag{2.4.24}$$

with $rand[0,1]$ a random number drawn by a uniform distribution between 0 and 1. An employer bee $\boldsymbol{x}_k$ (a candidate solution of the population) is associated to each food source $\boldsymbol{x}_i$ and it has to explore the neighbour of the local attractor according to the following equation defining a new artificial position $\boldsymbol{y}_i$:

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \quad i \neq k \tag{2.4.25}$$

where $\phi_{ij}$ is a random value between $-1$ and 1 and the indexes of the components are randomly chosen to respect the condition $i \neq k$. If $\boldsymbol{y}_i$ has a lower OF it will replace $\boldsymbol{x}_i$. The employer bees communicate the results of their local exploration to the other bees in the hive which is randomly initialized in a point of the design space. In the hive, there are present the onlooker bees which, after the news received by the employer bees, decide which food source to explore with a certain probability based on the fitness of the food sources. The scout bees are a third different kind of artificial bees which randomly explore the neighbour of the hive to verify the presence of better points close to the hive respect to the food sources [16]. The ABC algorithm was also adopted to solve structural optimization problems of trusses; for furthermore readings, one can refer for instance to [75, 16].
In 2010, the behaviour of bats inspired Xin-She Yang in 2010 to develop the Bat algorithm (BA). During the hunting phase, bats emit in 5-20ms ultrasonic pulses with a frequency range of 25-150 kHz to detect, throughout the returned echoes,

the external environment with a resolution of few millimeters, in the same order of magnitude of insects they are hunting for [82]. This echolocation mechanism was adapted in order to heuristically solve engineering problems adopting an emission rate and a loudness parameter to control the exploration and the exploitation of the algorithm.
All of the previous examples can be considered both nature-inspired algorithms, because the adoption of their biologically inspired rules, but also swarm-based algorithms because of the adoption of some concepts like social interaction and communication [82]. One example of Nature-inspired only is the Flower Pollination algorithm (FPA) proposed by Xin-She Yang based on the biological studies on the mechanism of biotic and abiotic pollination of some species of flower and plant which adopt Lévy flights and random walk principles [82].

### 2.4.3 Swarm-based Algorithms

The *swarm-based algorithms* are based on the collective behaviour of some animals which communicate and interact in order to find the food source in the natural environment. As stated in [82], the swarm-based algorithms started around 1990s with Marco Dorigo which proposed in his PhD dissertation a first example of ant colony optimization (ACO). In this case, the insects aggregate in colonies of huge size about from 2 to 25 millions of individuals [82]. Despite their dimensions, the experts observed that these insects have an emerging well-organized and cooperative global behaviour and they adopt local interactions by scent or trail of chemicals or pheromone and local rules which govern their movements in order to find the shortest or, in general, the best path to a food source [64]. In that sense, ACO tries to mimic the foraging search of these insects in order to solve many scheduling problems and finding the optimal route. When an agent passes on a specific path it releases a certain pheromone quantity. Then other insects will probably follow this specific path if the pheromone quantity is larger than other possible paths. To take into account a real situation such as the flow of time, some evaporation rules are adopted to decrease the concentration of the pheromone in time in order to abandon some possible not optimal routes. According to [64], ACO is suitable to solve discrete optimization problems. As a matter of fact, the optimization problem can be represented as a multi-layered graph in which the number of layers is equal to the number of design variables to optimize and the number of nodes in each layer is equal to the discrete permitted values for each design variable. In that sense, when an ant defines a specific path, the value of the node crossed by the ant defines the design vector (candidate solution). For a better clearness, one can refer to the Figure 2.7 in which the $k$-th ant of the colony defines a specific path from the nest to the food source. The values of the variable (nodes) which have been crossed by the ant in each layer form the design vector $\boldsymbol{x}_k$. Many variants of ACO are given by adopting different laws for pheromone release and its dynamic evaporation during the iterations. Two commonly used laws are incremental and exponential decay respectively adopted for the pheromone release and its evaporation [82]. In

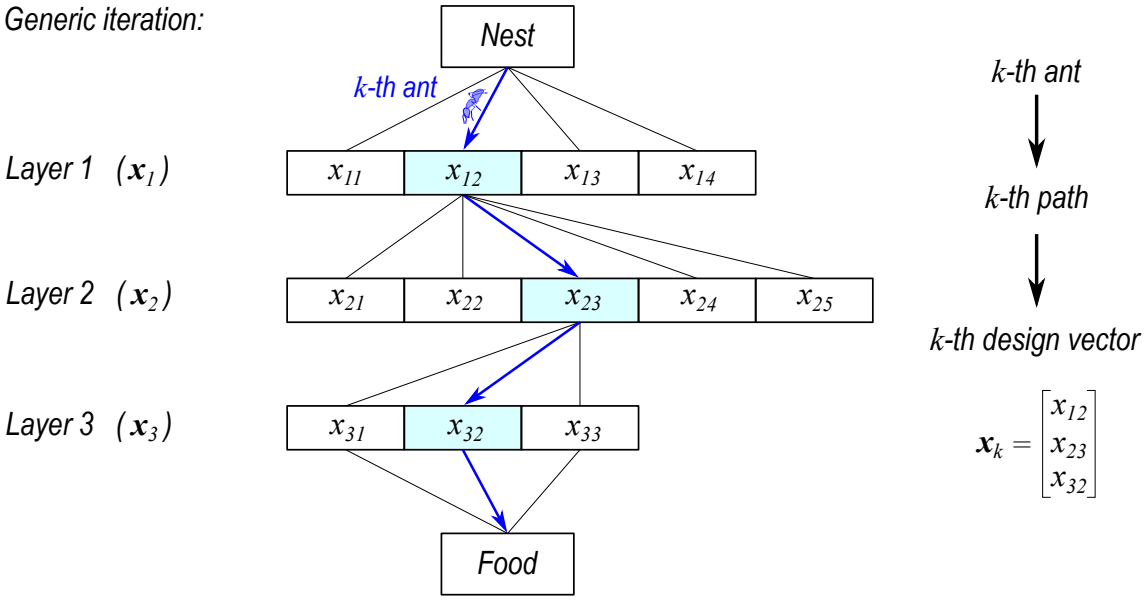



**Figure 2.7:** Representation of the working principles of the ACO with a problem with three discrete design variables (three layers). On the left, at a generic iteration different by the first one, the different arcs between two layer have different probability to be chosen. The $k$-th ant, in this case, identifies a path with a certain probability (blue lines) which will be conditioned by its trail pheromone. The values of the nodes crossed by the ant define the $k$-th design vector, which is presented on the right part of the image. This image is inspired by [64].

[64] the mechanism of the ACO applied to optimization problems it is analyzed. In particular, at the first iteration, all the paths are initialized with an equal dose of pheromone, thus they have the same probability to be chosen. A colony of $N$ ants start to choose each path from the nest till the end node where the ideal food source is located. The $k$-th ant at the $i$-th node of the layer $P$, chooses a specific $j$ node in the next layer $Q$ with a certain probability $p_{ij}$ which depend on the concentration $c_{ij}$ of the scent pheromone [82]

$$p_{ij} = \frac{c_{ij}^{\alpha} \cdot d_{ij}^{\beta}}{\sum_{j=1}^{n_Q} c_{ij}^{\alpha} \cdot d_{ij}^{\beta}}, \tag{2.4.26}$$

where $\alpha > 0$ and $\beta \geq 0$ are called influence parameter or degree of importance of the pheromones, $n_Q$ is the total number of nodes at the next layer $Q$ and $d_{ij}$ represent a parameter of desirability of the route. For example, this latter can represent the length of the arc between the node $i$ and $j$ when it is important to find the shortest path. The simplest case is to consider only the concentration as in [64] and this means setting $\beta = 0$. It is important to remember that an ant moves always forward until it reaches the end node. Afterwards, in a hypothetical cyclic way, a new iteration begins and the ants start again from the nest toward the end node and so on. When the $k$-th ant reach the end node, the path it has followed was subjected to an increase of the trail pheromone concentration. This amount, in each arc of the travelled path, is $\Delta c_{ij}^{(k)}$ and it is given by a local update rule which take into account e.g. the the

total length of the path $L_k$ of the $k$-th ant,

$$\Delta c_{ij}^{(k)} = \begin{cases} \dfrac{Q}{L_k)}, & \text{if } (i,j) \in \text{path of the ant } k, \\ 0 & \text{otherwise,} \end{cases} \tag{2.4.27}$$

where $Q$ is a fixed constant. It can be useful when the problem ask to find the shortest path. Instead, since one single path globally define a candidate solution of the optimization problem (a design vector $\boldsymbol{x}_k$), the update rule can take into account the fitness of the $k$-th design vector [28]:

$$\Delta c_{ij}^{(k)} = \begin{cases} \dfrac{Q}{f(\boldsymbol{x}_k)}, & \text{if } (i,j) \in \text{path of the ant } k, \\ 0 & \text{otherwise,} \end{cases} \tag{2.4.28}$$

It is important to notice that only the arc crossed by the $k$-th ant will rise its concentration of pheromone. In order to promote the exploration of new paths leaving the worse old paths, an evaporation rule is also considered. As proposed in [64] the residual concentration after the evaporation $c_{ij}^{(res)}$ is given by the following relation which is governed by a fixed evaporation rate, also known as pheromone decay factor, $\rho \in (0, 1]$,

$$c_{ij}^{(res)} = (1 - \rho) c_{ij}, \tag{2.4.29}$$

where $c_{ij}$ is the pre-existing concentration value in the arc $(i, j)$ without considering any new increment. Since the number of the ant of the colony is $N$, the single arc $(i, j)$ will be subjected to a global variation of concentration equal to the old concentration reduced by the evaporation rule and the sum of the increment of concentration of each ant which passed in that arc. In symbols the concentration $c_{ij}^{(new)}$ of the arc $(i, j)$ at the next iteration is

$$c_{ij}^{(new)} = c_{ij}^{(res)} + \sum_{k=1}^{N} \Delta c_{ij}^{(k)} = (1 - \rho) c_{ij} + \sum_{k=1}^{N} \Delta c_{ij}^{(k)}. \tag{2.4.30}$$

In [64] it is proposed an update rule for concentration variation which take into account the best and the worst path among all the ants of the colony:

$$\Delta c_{ij}^{(k)} = \begin{cases} \dfrac{\zeta f_{best}}{f_{worst}} & \text{if } (i,j) \in \text{global best path} \\ 0 & \text{otherwise,} \end{cases} \tag{2.4.31}$$

where $\zeta$ is a scale factor which governs the amount of pheromone released in function of the ratio between the best OF $f_{best}$ and the worst OF $f_{worst}$ among all the paths. This latter pheromone update rule aims to promote the exploitation supporting the path which lead to the best OF.
As usual, the stopping criterion can be set when the OF stalls for a prescribed number of iterations or when a certain maximum number of allowed iterations is reached [64].

Later, in 1995 James Kennedy and Russell C. Eberhardt, [34] proposed the most popular swarm intelligent algorithm in the intelligent computing field, the particle swarm optimization (PSO). It was inspired by the metaphor of the collective and social behaviour of communities of individuals called swarms such as bird flocking, fish schooling, or swarming of insects with an emergent intelligent and structured system. A specific insight of this topic is presented in the next chapter because it is the base algorithm adopted in the experimentation of a new technique in constraint handling proposed in the present Thesis.

# Chapter 3

# Particle Swarm Optimization Algorithm

Within the framework of soft computing methodologies, a large number of non-conventional paradigms have been explored in order to create efficient and user-friendly optimizers. Nowadays, a wide variety of biological, social and physical metaphors has been analyzed and tested. In the present Thesis, the class of optimizers based on swarm intelligence, the so-called Particle Swarm Optimization algorithms are considered, which have been proposed in 1995 by James Kennedy and Russell C. Eberhardt, [34]. To make some first comparisons with EAs we can refer to [62]. EAs are based on simulation of natural Darwin's theory evolution process with the survival of fittest members, whereas the swarm intelligence is based on a collective behaviour in which each element move independently in search space. Thanks to somehow interaction among members of community, the entire swarm shows an intelligent global behaviour moving toward the optimal solution. This emerging class of optimizers is inspired by social behaviours observable in certain natural aggregations, such as bird flocking, fish schooling, or swarming of insects when they search for food, resources or protection. Every member of the population searches in its neighbourhood for the best outcome, learns from its own experience as well as from the other members' findings. Typically, if a member of the swarm discovers a desirable path to go, then the rest of the swarm will follow quickly. Adopting the AI notation, the swarm population is composed of simple intelligent search agents whose movements it is not governed from an external control unit but they interact each other and with the environment to realize a collective intelligent global behaviour [62]. Thus, similar to other EAs, a PSO is a population-based optimizer and can solve complex non-convex optimization problems. PSO is based on the principle that social sharing of information among the individuals of the population can lead to optimum solutions. In fact, as affirmed in [59], every particle possesses a memory of the best position it has visited. Hence an appropriate combining of the self-experience of every particle with the global best position of the entire swarm, we can find a balance between exploration and exploitation. It is a quite simple algorithm since it requires few lines of code in any programming language and a small set of parameters have to be defined and tuned. PSO is not complicated, resulting in an attractive tool for non-experts in the field of evolutionary computation. Several

studies (e.g., Kennedy and Eberhart [35]) demonstrated that this optimizer has a good convergence rate.

## 3.1 PSO framework

Based on the swarm intelligence theory, two different categories of PSO optimizers can be formalized:

- PSO algorithms in which it is assumed that a Newtonian dynamics regulates the movement of the particles.
- Quantum-behaved (Q-PSO) algorithms, in which the Newtonian hypothesis is rejected. In this case, the usual metaphors for PSO are replaced with physical paradigms related to the movement of particles in the atomic or sub-atomic scale. Thus, the classical mechanic approach adopted for representing the dynamics in traditional PSO is replaced with the quantum mechanics process where the term "trajectory" is meaningless.

In this thesis, the first class of PSO optimizers is considered in the final practical applications, in which according to Newton's theory, both position and velocity of the swarm can be determined simultaneously.

In the general formulation of PSO, also called *canonical formulation* [43], the $i$th particle ($i = 1, ..., N$, where $N$ denotes the population size) at iteration $k$ has two attributes, that are its velocity ${}^{k}\mathbf{v}_i = \{{}^{k}\mathrm{v}_{i1}, ..., {}^{k}\mathrm{v}_{ij}, ..., {}^{k}\mathrm{v}_{in}\}$ and position ${}^{k}\boldsymbol{x}_i = \{{}^{k}x_{i1}, ..., {}^{k}x_{ij}, ..., {}^{k}x_{in}\}$. To protect the cohesion of the swarm, the velocity ${}^{k}\mathrm{v}_{ij}$ is forced to be (in absolute value) less than a maximum velocity $\mathbf{v}_j^{\max}$ with $\mathrm{v}^{\max} = \{\mathrm{v}_1^{\max}, ..., \mathrm{v}_j^{\max}, ..., \mathrm{v}_n^{\max}\}$. Typically, it is assumed that $\mathrm{v}^{\max} = \gamma(\boldsymbol{x}^u - \boldsymbol{x}^l)/\tau$, in which the time-related parameter $\tau = 1$ is introduced to assign a physical meaning to the formula and $\gamma$ defines how far a particle can move starting from its current position [62]. Nevertheless, there is not sufficient degree of uniformity about the choice of $\gamma$ whose numerical value can vary significantly, usually in the range $[0.1, 1]$ and generally it is set to $\gamma = 0.50$ [62]. Since there are usually not additional information about the optimization problem to deal with (black-box optimization problem [43]), the initial values ${}^{0}\boldsymbol{x}_i$ for $i = 1, ..., N$ are derived by generating pseudo-randomly the collection of $N$ solutions within the assigned search space. Moreover, ${}^{0}\boldsymbol{v}_{ij}$ is pseudo-randomly generated using a uniform distribution between $-\mathbf{v}_j^{\max}$ and $+\mathbf{v}_j^{\max}$. For this purpose, the Latin Hypercube Sampling (LHS) technique has been iteratively used to generate the best initial population with minimum correlation between samples (see also Monti et al. [49]). According to [59], making a comparison with EAs, it is possible to interpret the randomness in the setting of velocity particles as a "directional mutation operator". At iteration $k + 1$ the velocity ${}^{(k+1)}\mathrm{v}_i$ and the position ${}^{(k+1)}\boldsymbol{x}_i$ vectors are evaluated as follows

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{v}_i + c_1\, {}^{(k+1)}\boldsymbol{r}_{1i} * \left[{}^{k}\boldsymbol{x}_i^{Pb} - {}^{k}\boldsymbol{x}_i\right] + c_2\, {}^{(k+1)}\boldsymbol{r}_{2i} * \left[{}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i\right], \tag{3.1.1} $$

$$ {}^{(k+1)}\boldsymbol{x}_i = {}^{k}\boldsymbol{x}_i + \tau\ {}^{(k+1)}\mathbf{v}_i \quad (\tau = 1). \tag{3.1.2} $$

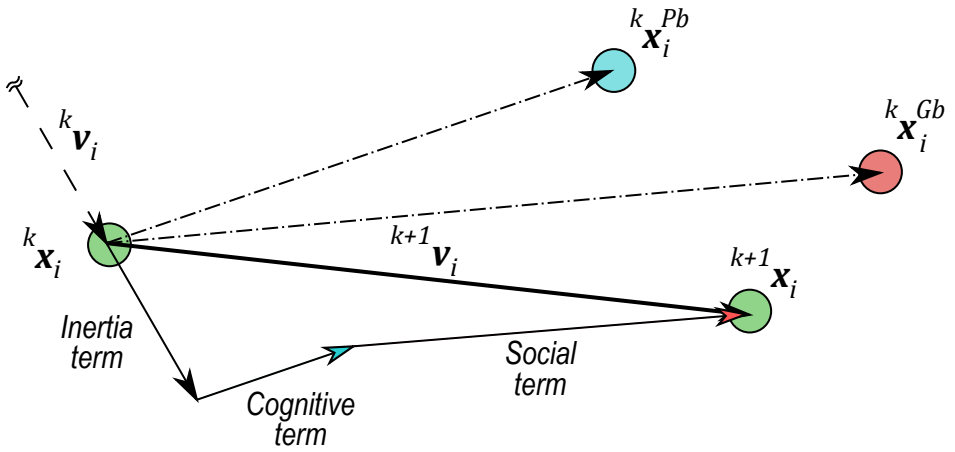



**Figure 3.1:** Graphical representation of the standard Newtonian PSO mechanism. The three main contribution of the velocity vector ${}^{k+1}\mathbf{v}_i$ stated in (3.1.1) for the $i$-th particle are depicted: the *Inertia term* is related to the Newtonian dynamic-based mechanism and depends on the previous velocity vector ${}^{k}\mathbf{v}_i$; the *Cognitive term* is related to the attraction of the Pbest, ${}^{k}\boldsymbol{x}_i^{Pb}$ (best position visited by the particle up to iteration $k$); the *Social term* is related to the attraction of the Gbest, ${}^{k}\boldsymbol{x}_i^{Gb}$ (best position visited by the entire swarm up to iteration $k$).

where ${}^{k}\boldsymbol{x}_i^{Pb}$ is the best previous position of the $i$th particle (also known as *pbest*)

$$ {}^{k}\boldsymbol{x}_i^{Pb} = \begin{cases} {}^{k}\boldsymbol{x}_i, & \text{if } f({}^{k}\boldsymbol{x}_i) < f({}^{(k-1)}\boldsymbol{x}_i^{Pb}), \\ {}^{(k-1)}\boldsymbol{x}_i, & \text{otherwise.} \end{cases} \tag{3.1.3} $$

given that ${}^{0}\boldsymbol{x}_i^{Pb} = {}^{0}\boldsymbol{x}_i$. The term ${}^{k}\boldsymbol{x}^{Gb}$ is called *gbest* and it is the best performer among the entire swarm or among a set of particles belonging to a certain neighbourhood. According to the adopted definition for the neighbourhood, one can obtain different schemes of PSO which are analyzed in the next Section of the present Thesis. Supposing now that all the particles are able to share information with each other about the best performer of the swarm, the gbest can be obtained adopting the following relation

$$ {}^{k}\boldsymbol{x}^{Gb} = \arg\min_{i}\{f({}^{k}\boldsymbol{x}_i)\}. \tag{3.1.4} $$

The acceleration factors $c_1$ and $c_2$ in (3.1.1) (both positive scalars) are called *cognitive* and *social* parameters, respectively. Moreover, ${}^{(k+1)}\boldsymbol{r}_{1i}$ and ${}^{(k+1)}\boldsymbol{r}_{2i}$ are vectors whose terms are pseudo-random numbers uniformly distributed in the interval $[0, 1]$, while the symbol $*$ denotes the term-by-term vector multiplication (Hadamard product [59]). A graphical representation of the PSO velocity update mechanism stated in (3.1.1) is depicted in Figure 3.1. Adopting the AI notation, it is possible to state that the intelligent search agents exhibit a stochastically biased movements influenced by the pbest and gbest which lead to a global convergent behaviour of the swarm [43]. The superscripts on the left and the subscripts on the right denote that a different couple of pseudo-random vectors is needed for each particle at any iteration. It should be mentioned at this point that the formulation (3.1.1) is rather uncommon, in the sense that the majority of researchers in the field adopt a unique

set of pseudo-random terms for any dimension of the search space. Nonetheless, Liang et al. [40] pointed out that Eq. (3.1.1) yields better performance because of its problem-invariant property. The check on the maximum admissible velocity for any particle i is performed at iteration k in the following manner

$$ {}^{k}\mathrm{v}_{ij} = \begin{cases} \mathrm{sign}[{}^{k}\mathrm{v}_{ij}]\mathbf{v}_j^{\max}, & \text{if } \left|{}^{k}\mathrm{v}_{ij}\right| > \left|{}^{k}\mathbf{v}_j^{max}\right|, \\ {}^{k}\mathrm{v}_{ij}, & \text{otherwise} \end{cases} \qquad \forall j = 1, ..., n. \tag{3.1.5} $$

where sign[·] is the sign operator. Another check is needed to verify that the particle is within the feasible search space, respecting the side constraints,

$$ \left({}^{k}x_{ij}, {}^{k}\mathrm{v}_{ij}\right) = \begin{cases} \left({}^{k}x_{ij}, {}^{k}\mathrm{v}_{ij}\right), & \text{if } x_j^l \leq {}^{k}x_{ij} \leq x_j^u, \\ \left({}^{k}x_{ij} = x_j^l, {}^{k}\mathrm{v}_{ij} = 0\right), & \text{if } {}^{k}x_{ij} < x_j^l, \\ \left({}^{k}x_{ij} = x_j^u, {}^{k}\mathrm{v}_{ij} = 0, \right), & \text{otherwise.} \end{cases} \tag{3.1.6} $$

The infeasible particles' velocity is fixed to zero in (3.1.6) for the next iteration to avoid considering any points outside the search space. Following iteratively this simple set of instructions, the swarm is expected to "fly" towards the global optimum of the problem. Since the required number of iterations $L$ is not known *a priori* and therefore a stopping criterion is needed. In general, stopping criteria in PSO can be similar to those typically adopted for several EAs, see for instance [50] and its references. In this study, the search is terminated once a maximum number of iterations L is achieved. Although this strategy has the disadvantage to require some information about the problem or some preliminary runs, it appears to be useful when some parameters of the optimizer have to be iteratively tuned during the process. The interested reader is referred to the work by Li and Xiao [37] for a useful discussion on the selection of the number of iterations for PSO.

The performance of PSO strongly depends on choosing control parameter values, see Quaranta 2020 [62]. Firstly, although it might seem better to choose swarm size $N$ as bigger as possible it would lead to a very slow algorithm. Moreover, its choice should be based on the number of design variables $n$, but it has been experimentally demonstrated which there is no substantial difference when $N$ varies in the range $[20, 100]$ for a maximum number of design variables $n_{\max} = 30$ [62]. Furthermore version of PSO called micro-PSO ($\mu$PSO) which can work also with very small swarm size was also developed in years. Inertia weight and acceleration factors are also control parameter values which can affect the performances. The use of inertia weights in PSO has been proposed by Shi and Eberhart [73], where the authors introduced this parameter in an effort to improve the convergence of the standard PSO. This concept is not new in soft computing community; actually, it is similar to the momentum term in a gradient descent artificial neural network training algorithm, or the temperature adjustment schedule for simulated annealing algorithms. Typical range of values for $w$ was $[0.8, 1.2]$. In a subsequent study by the same authors, a linearly decreasing inertia weight has been adopted [74]

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}w\, {}^{k}\mathbf{v}_i + c_1\, {}^{(k+1)}\boldsymbol{r}_{1i} * \left[{}^{k}\boldsymbol{x}_i^{Pb} - {}^{k}\boldsymbol{x}_i\right] + c_2\, {}^{(k+1)}\boldsymbol{r}_{2i} * \left[{}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i\right], \tag{3.1.7} $$

with

$$ {}^{k}w = ({}^{0}w - {}^{L}w)\frac{L-k}{L} + {}^{L}w \tag{3.1.8} $$

in which ${}^0w$ and ${}^Lw$ are the initial and the final values of the inertia weight, respectively. In principle, the inertia weight is a scaling factor of the previous velocity of the particle and its role is to control the exploration of the swarm: a large inertia weight will force larger velocity at the next generation and the swarm is expected to explore a larger region of the search space. In contrast, small inertia values have to be introduced to improve the local exploration. Some authors proposed also Non-Linear updating law for inertia weight e.g. in [59] define a three stages reduction of inertia weight using a cubic polynomial function in order to have fast reduction at initial stages and slower reduction at last iterations. One can check for further Non-Linear formulation about inertia weight in [71]. Concerning acceleration factors, some authors proposed varying models but usually they are assumed statically fixed to $c_1 = c_2 = 2$ [62]. In another version of PSO, inertia weight is not considered and it is replaced with a constriction factor $\chi$ which multiply the whole second member of the velocity expression (3.1.1):

$$ {}^{(k+1)}\mathbf{v}_i = \chi \left[ {}^{k}\mathbf{v}_i + c_1 \, {}^{(k+1)}\boldsymbol{r}_{1i} * \left( {}^{k}\boldsymbol{x}_i^{Pb} - {}^{k}\boldsymbol{x}_i \right) + c_2 \, {}^{(k+1)}\boldsymbol{r}_{2i} * \left( {}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i \right) \right]. \quad (3.1.9) $$

The constriction factor expression is the following [62]

$$ \chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|}, \quad \text{with } \varphi = c_1 + c_2 > 4. \quad (3.1.10) $$

According to [62], typically it is assumed $\varphi = 4.1$ which implies $\chi = 0.729$ and, setting the same value to acceleration factor, it leads to $c_1 = c_2 = 2.05$. Although dynamic (deterministic and non-deterministic) models exist for social/cognitive factors and inertia weights, in this study it is assumed which all of them are constant values equal to $c_1 = c_2 = 2$, ${}^0w = 0.90$ and ${}^Lw = 0.40$ [57].

## 3.2 Historical Overview and main variants

The use of evolutionary and swarm intelligence algorithms is constantly gaining popularity and many complex optimum design problems have been efficiently solved using Nature-inspired memetic and meta-heuristic methods. In recent years, efficient optimizers based on swarm intelligence, namely, Particle Swarm Optimization algorithms, proposed originally by Kennedy and Eberhart in 1995 [34], have evolved.

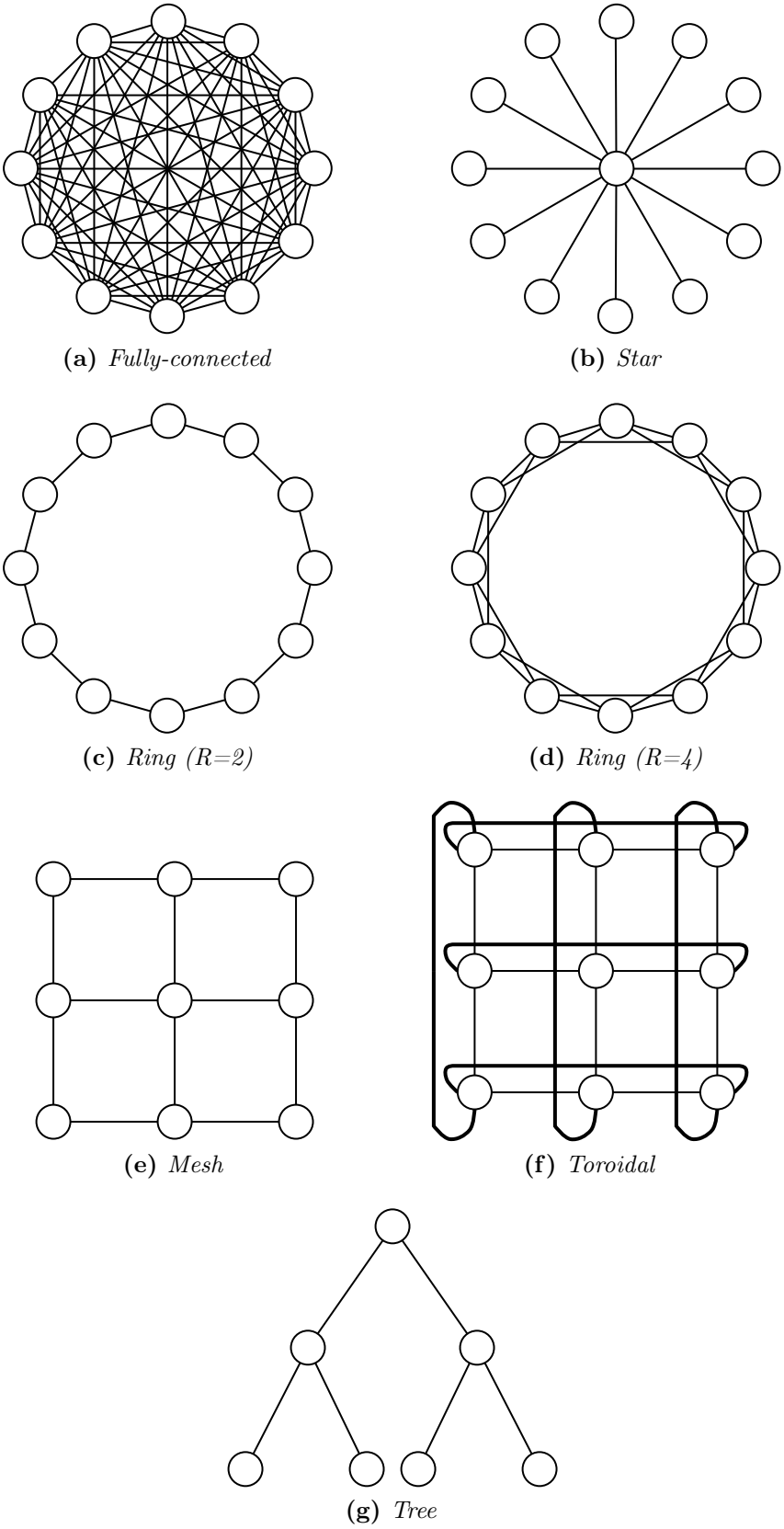
The canonical formulation of PSO posed immediately some issues related to the choice of the hyper-parameters because, in practice, every prior user-defined decision strongly affect the behaviour and the convergence of the PSO [43]. As one can notice from the velocity rule update (3.1.1), for the best particle gbest is equal to pbest and so the cognitive and the social terms can be combined together. The velocity vector for the best particle in the early formulation took the following form

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{v}_i + c \left[ {}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i \right], \quad (3.2.1) $$

were $c = c_1 + c_2$. If $c_1 + c_2 \geq 4$, the early version was subjected to an excessively increase of the velocity vector which lead to the so called *swarm explosion effect* [43]. For this reason the first improvements posed a *velocity clamping*

($-\mathbf{v}_j^{\max} \leq {}^{(k+1)}\mathbf{v}_j \leq +\mathbf{v}_j^{\max}$) and the *inertia weight* term to avoid this problem, as already explained in the previous Section. Afterwards, in 2002 M. Clerc and J. Kennedy performed the first convergence and stability analysis of PSO assuming the early formulation with 1-dimensional particles without stochasticity but considering the velocity clamping [43]. The outcome of the study were a proposal of the optimal hyper-parameters of the constrinction factor and the social and cognitive factors, obtained in a closed form. Therefore, they suggest to adopt $\chi = 0.729$ and $c_1 = c_2 = 1.49$ when PSO is implemented and, nowadays, it is still a common solution [43].

Later, the attention was focused on the neighbourhood concept. In fact, it represents the way in which the information are transmitted among the particles. Without an efficient exchange of these information, the swarm can not exhibit the collective convergent behaviour. In particular, the structure of the neighbourhood, also known as *neighbourhood topology*, defines the way in which the particles are interconnected and, therefore, how the information are channelled among the particles, whereas the size of the neighbourhood affect the influence of the swarm on each particle [43]. If ${}^{k}\boldsymbol{x}^{Gb}$ denotes the best position among all the particles in the swarm, the so called *gbest* as defined in (3.1.4), then the swarm is denoted as fully informed or fully connected. When the PSO adopt this scheme, it is named as global PSO model or simply *gbest model* [43, 62, 71]. This is the most popular time-invariant neighbourhood because of its ease and rapid implementation. As one can see in the Figure 3.2 (a), in this topology scheme, all the particles influence each other in a fully connected net which brings to speed up convergence rate toward the global best particle position. However, it is even more probable that the swarm could be entrapped in a local minimum due to the high influence rate on the entire swarm of the entrapped particles [43]. Therefore, some local strategies were implemented in order to reduce the size or even change the structure of the neighbourhood. These strategies are denoted as local PSO models or simply *lbest models* [43, 62, 71]. These methods can be implemented considering that each particle in the numerical vector has a unique index, therefore each particle can unequivocally be selected to enter in a neighbourhood through its index [44]. The easiest to be implemented local PSO neighbourhood is the ring topology, Figures 3.2 (c) and (d). As explained in [44], this topology forms a neighbourhood considering the nearest indexes of the particle under consideration, resulting in an ideal circular interconnection. The amount of the particles which enter in the neighbourhood considered is called radius $R$. In the Figure 3.2 (c) the ring neighbourhood with radius $R = 2$ is showed, whereas in Figure 3.2 (d) there is the ring neighbourhood with radius $R = 4$. The resulting scheme create small sets of neighbourhood for each particle and, therefore, it slows down the information transmission. In fact, when a certain neighbourhood find the global optimum, it has to communicate to their nearest neighbourhood which in turn will communicate to the others and so on. Another possible way, it is the star, focal or wheel neighbourhood [70] presented in the Figure 3.2 (b). In this topology, all the information have to pass through a central focal particle which can be selected randomly in the population. This central particle acts as a filter or buffer for the information transmission. As a matter of fact, all the particles tend to follow the central one which in turn is attracted from the best ones of the entire swarm. The



**(a)** *Fully-connected* **(b)** *Star*

**(c)** *Ring (R=2)* **(d)** *Ring (R=4)*

**(e)** *Mesh* **(f)** *Toroidal*

**(g)** *Tree*

**Figure 3.2:** Some examples of PSO Neighborhood Topologies

resulting scheme enhance the exploration to avoid the entrapping in local minima but it slows down the convergence rate [44]. In the mesh topology, the particles are organized in a quite different way because they assume an idealized lattice scheme with a particle at each vertices. This scheme is graphically depicted in Figure 3.2 (e). Every central particle is connected to the four nearest particles, but the boundary particles are connected only with three and the corner ones only with two. A little variation of this scheme leads to the toroidal topology, also called grid or von Neumann topology in [70]. As shown in Figure 3.2 (f), to avoid the unbalance of the neighbourhood size of the boundary and corner particles of the mesh scheme, these particles are connected to the symmetric corner or boundary particles. In this way, every particle is connected to four particles and this leads to an idealized three-dimensional scheme. This latter reminds the geometrical closed surface torus and this explains why this neighbourhood is also called Clifford torus [44]. As stated in [70], it is also possible to adopt a random neighbourhood topology defined adopting a certain distribution. A completely different neighbourhood topology scheme is the tree topology. This is based on a hierarchical scheme where, at the first layer, a random particle is selected as a root node. This latter is connected to two child particles (binary tree) and it can communicate with them in order to find the best attractor. Every child particle can communicate with another sub-layer of two child particles but also with the parent particle. As depicted in Figure 3.2 (g), at the same layer the particles can not communicate each other but the information have to pass to the parent node and, afterwards, they can be transmitted to the other child particle. As stated in [44], the binary resulting tree has to be symmetrical in order to avoid unbalances. All the previous neighbourhood topologies are called static topologies in [70] when they remain fixed for all the iterations, whereas they are denoted as dynamic when they change during the iterations. In fact, starting with lower size topologies can improve the exploration and, afterwards, enlarging the size of the neighbourhood until it reduces to the fully connected model, it can enhance the exploitation and the convergence speed rate. Another dynamic example consists in dividing the swam in small random neighbourhoods which are periodically randomly reassembled, as proposed in [70]. As demonstrated in [44], better convergence results were obtained adopting the lbest model respect to the gbest model, in particular the widely used ring topology, because it slows down the convergence speed rate but avoiding the premature convergence. In fact, as stated in [43] the neighbourhood topology has to be carefully defined in order to balance the exploration and the exploitation capabilities of the algorithm.

Since there are usually not prior available information about the optimization problem to solve, which is called black-box optimization problem, the random initialization of the particles' positions remains the standard procedure. If some information are available, there are many variants which introduce other initialization models or simply modify the probability distribution used in the random initialization to biased it in a certain manner [43]. As proposed in [70], another possible solution is the initialization of the swarm with all the velocities set to zero or even considering random pairs of particles and posing the starting velocity as the mean of the two considered positions. For the termination criteria, there are also many

variants because it is possible to define a stagnation criterion when all the particle collapse in a certain point for a certain number of iterations, or posing some limitation on the computational time detecting the number of OF evaluations [43].

According to [70], some other variants proposed to improve the self-adaptation of the PSO introducing a time-varying parameter adaptation in which the constriction factor $\chi$, the cognitive factor $c_1$ and the social factor $c_2$ dynamically change during the iterations. For example, a simple approach is to set a linear decreasing law for $\chi$ with the increase of the iteration number. In order to enhance the exploration at the beginning and the exploitation at the end, $c_1$ is set with a decreasing law, whereas $c_2$ with an increasing law with the proceeding of the iterations.

Some specific techniques were proposed to deal with specific class of problem which aim to transform the OF in order to remove the local optima points. For instance, the stretching technique consists of a global transformation through a filled function which detect the local minima points and tries to alleviate the OF in those points. Unfortunately, this can also lead to the introduction of new local minima and also to the Mexican hat effect. Therefore the deflection technique act similarly to the stretching but only locally around the local minimum detected [43].

From the historical point of view, [43] provides an overview on the main variants of the 1995 canonical PSO by Kennedy and Eberhardt. In 1997, a binary encoding variant was proposed in the so called Binary PSO (BPSO). In this variant, the $j$-th component of the position of the $i$-th particle is defined as

$$ {}^{(k+1)}x_{ij} = \begin{cases} 1 & \text{if } rand[0,1] < S\left({}^{(k+1)}v_{ij}\right), \\ 0 & \text{otherwise,} \end{cases} \tag{3.2.2} $$

where $S({}^{(k+1)}v_{ij})$ denotes that the velocity vector is transformed into a moving probability through a sigmoid function which is defined as

$$ S(x) = \frac{1}{1+e^{-x}}. \tag{3.2.3} $$

In 2002 researchers proposed the Guaranteed Convergence PSO (GCPSO) because they noticed an issue that prevent the convergence with the gbest model equipped with inertia weight strategy. In fact, when the particle position coincides with the pbest and the gbest, the velocity update rule relies only on the inertia term. If the velocity tends to be zero, this results in a immobility of the particle and the gbest position stalls and can no more improve. Therefore, researchers proposed a different update rule of the gbest particle adding some randomness and a dynamically defined scaling factor which try to refine the optimal solution and to address the stagnation problem [43].

According to [70], in the same year, Kennedy proposed also two other versions of PSO called social only PSO, in which the cognitive factor $c_1$ was zeroed, and the cognition-only PSO in which, on the contrary, no gbest attractor was adopted, setting $c_2$ to zero.

As reported in [44], in 2002 Clerc and Kennedy analytically proved that each particle converges to the average weight of the global attractor and the cognitive best visited position

$$\lim_{k\to+\infty} {}^{(k+1)}x_{ij} = \frac{c_1\,{}^{(k+1)}x_{ij}^{Pb} + c_2\,{}^{(k+1)}x_{ij}^{Gb}}{c_1 + c_2}. \tag{3.2.4}$$

The above weighted mean highlight that the canonical PSO strongly depends on the parameter choice. This led researchers to try to find new variants in order to reduce the parameter dependency and, in 2003, the Bare Bones PSO (BBPSO) was introduced by Kennedy [44]. This variant replaces the velocity and the position update rules stated in (3.1.1) and (3.1.2) with a random Gaussian extraction around the best positions:

$${}^{(k+1)}x_{ij} \sim \mathbf{N}\left(\mu_{ij}(k), \sigma_{ij}^2(k)\right), \tag{3.2.5}$$

where the mean value and the variance are respectively equal to

$$\mu_{ij}(k) = \frac{{}^{(k)}x_{ij}^{Gb} + {}^{(k)}x_{ij}^{Pb}}{2}, \qquad \sigma_{ij}^2(k) = \left|{}^{(k)}x_{ij}^{Gb} - {}^{(k)}x_{ij}^{Pb}\right|. \tag{3.2.6}$$

As stated in [43], in 2004, a variant named as Fully Informed PSO (FIPS) extended the concept of neighbourhood influence to a global view. As a matter of fact, respect the canonical version in which the velocity update rule only depends on the cognitive and the global attractors, in this variant the attraction of the all defined neighbourhood is considered. In the FIPS the two equations (3.1.1) and (3.1.2) are replaced by

$${}^{(k+1)}x_{ij} = {}^{(k)}x_{ij} + \chi\left({}^{(k)}x_{ij} - {}^{(k-1)}x_{ij}\right) + \sum_{q\in NB_i} C_q\left({}^{(k)}x_{qj}^{Pb} - {}^{(k)}x_{qj}\right), \tag{3.2.7}$$

where the sum is conducted on all the elements belonging to the neighbourhood of the $i$-th particle denoted as $NB_i$, and $C_q$ is a random coefficient drawn from a uniform distribution $\mathbf{U}[0, c/s_i]$ with $c = c_1 + c_2$ as usual and $s_i$ is the neighbourhood size.

According to [43], always in 2004, a new variant of PSO, the Quantum PSO (QPSO), drastically changed the framework abandoning the Newtonian dynamics laws for a quantum mechanics governed environment. In a Newtonian field, both the position and the velocity of a particle can be defined simultaneously, but it is not true in quantum mechanics because of the uncertainty principle in which the term "trajectory" is meaningless. As a matter of fact, a particle possesses a quantum state which can be defined by a wave function $\Psi(\boldsymbol{x}, t)$ which depends on the position of the particle and the time. This is a state function for a system and it is connected to the probability density function ($|\Psi|^2$) to find a particle in a elementary volume ($\mathrm{d}x\mathrm{d}y\mathrm{d}z$) in a certain position of the physical space $(x, y, z)$. At the microscopic level, the time-dependency properties of the particles are governed by the Schrödinger equation [78]. The first QPSO proposal assumed that the state of a PSO particle $\boldsymbol{x}$ in the quantized search space was governed by the Delta potential well function, with center in the point $\boldsymbol{p}$ [78]:

$$\Psi = \frac{1}{L} e^{-\|\boldsymbol{p}-\boldsymbol{x}\|/L}, \tag{3.2.8}$$

where the term $L$ is defined below. This assumption allows the swarm to converge to $\boldsymbol{p}$ without the explosion of the swarm [77]. In the quantized search space, it is possible to know only the probability that a particle appears in a certain region but, in order to perform the OF evaluation, the position is an absolutely necessary information. Therefore, the quantized space is transformed into a classic solution space to make possible the measurement of the particle position [77]. As stated in [78], they adopt the Monte Carlo Method to numerically simulate the process of measurement finding the positions of the particles with a sufficient accuracy. Therefore, in the original QPSO formulation, after randomly initialized the swarm population, and after defining the gbest particle in a fully informed neighbourhood, the next position ${}^{(k+1)}\boldsymbol{x}_i$ is given by the following relation

$$ {}^{(k+1)}\boldsymbol{x}_i = \begin{cases} {}^{(k)}\boldsymbol{p}_i - L \cdot \ln\left(1/u\right), & \text{if } rand[0,1] > 0.5, \\ {}^{(k)}\boldsymbol{p}_i + L \cdot \ln\left(1/u\right), & \text{otherwise,} \end{cases} \tag{3.2.9} $$

where $u = rand[0,1]$ and, considering the converge of the PSO particles (3.2.4), the center of the Delta well function, ${}^{(k)}\boldsymbol{p}_i$ can be assumed as

$$ {}^{(k)}\boldsymbol{p}_i = \frac{\varphi_1 \, {}^{(k)}\boldsymbol{x}_i^{Pb} + \varphi_2 \, {}^{(k)}\boldsymbol{x}_i^{Gb}}{\varphi_1 + \varphi_2}, \tag{3.2.10} $$

with $\varphi_1, \varphi_2 = rand[0,1]$ and the parameter $L$, called creativity or imagination of the particle in [77], is given by

$$ L = \beta \cdot \left\| {}^{(k)}\boldsymbol{x}_i - {}^{(k)}\boldsymbol{x}_i^{Pb} \right\|, \tag{3.2.11} $$

where $\beta$ is a positive user-defined control parameter called creativity coefficient in [77] or contraction–expansion coefficient in [62]. A revised QPSO was later proposed by the same authors modifying the (3.2.11) as below to improve the convergence rate and the performance of the algorithm,

$$ L = \beta \cdot \left\| {}^{(k)}\boldsymbol{x}_i - {}^{(k)}\boldsymbol{x}_i^{Mb} \right\|. \tag{3.2.12} $$

The new above definition of $L$ take into account not only the cognitive best visited position ${}^{(k)}\boldsymbol{x}_i^{Pb}$ but the mean best positions of the entire swarm ${}^{(k)}\boldsymbol{x}^{Mb}$ called *mbest*,

$$ {}^{(k)}\boldsymbol{x}^{Mb} = \sum_{i=1}^{N} {}^{(k)}\boldsymbol{p}_i / N, \tag{3.2.13} $$

where $N$ is the swarm size. As stated in [78], a particle has a certain probability $z$ to appear in a certain zone, therefore in order that the quantum state converges to the center $\boldsymbol{p}$ it is necessary that $z > 0.5$. This explains why the new position given by the (3.2.9) is defined by two different laws with a certain probability. In [78], to guarantee the convergence of the swarm towards the center $\boldsymbol{p}$, the limit of $L$ with $k \to \infty$ need to be zero. Therefore the creativity parameter $\beta$ was originally proposed to be $\beta < 1/\ln\left(\sqrt{2}\right)$. The main advantages of adopting the quantum PSO are related to the fact that the PSO was adapted to a complex non-linear system in

which the principle of superposition of the states holds and which is based on the uncertainty, which is quite different from the classical stochastic concept. In fact, as explained in [77], before the measurement phase, the particle can appear anywhere with a certain probability in the quantized search space because it does not follow the rule of a deterministic classical trajectory, thus allows a better exploration of the search space. Later, other main variants of the QPSO have been proposed adopting different mathematical form for the quantum mechanics potential field e.g. the Harmonic Oscillator function or the Square Well function [43].

In 2004, according to [43, 56], the Unified PSO (UPSO) tried to unified the gbest model with the lbest model defining the update velocity relation as

$$^{(k+1)}\mathrm{v}_{ij} = u\,{}^{(k+1)}G_{ij} + (1-u)\,{}^{(k+1)}L_{ij} \tag{3.2.14}$$

where $u \in [0,1]$ is called unification factor and govern the relative weight of the lbest model ${}^{(k+1)}L_{ij}$ defined as

$${}^{(k+1)}L_{ij} = \chi\,{}^{(k)}\mathrm{v}_{ij} + c_3({}^{(k)}x_{ij}^{Pb} - {}^{(k)}x_{ij}) + c_4({}^{(k)}x_{ij}^{NB_i} - {}^{(k)}x_{ij}). \tag{3.2.15}$$

where ${}^{(k)}x_{ij}^{NB_i}$ is the best attractor of the neighbourhood of the $i$-th particle, and the gbest model ${}^{(k+1)}G_{ij}$ defined as

$${}^{(k+1)}G_{ij} = \chi\,{}^{(k)}\mathrm{v}_{ij} + c_3({}^{(k)}x_{ij}^{Pb} - {}^{(k)}x_{ij}) + c_4({}^{(k)}x_{ij}^{Gb} - {}^{(k)}x_{ij}). \tag{3.2.16}$$

Two later versions introduced a stochastic parameter $r \sim \mathbf{N}(\mu, \sigma^2)$ to the ${}^{(k+1)}G_{ij}$ or to the ${}^{(k+1)}L_{ij}$ in (3.2.14) which simulates the mutation operator of the EAs.

In 2004, another PSO variant called Cooperative PSO (CPSO) was also presented [43]. This variant was based on a multi-population scheme in which each population performed a research only a smaller restricted region of the entire search space but they cooperated together in the combining phase to find the optimum.

Starting from 2005, the PSO was integrated with mathematical local search algorithms giving rise to the so called Memetic PSO (MPSO). For example, the gradient-based optimization methods were implemented to perform a refined search of the gbest found or in order to enhance the search in a specific direction [43].

Instead in 2006, the Comprehensive Learning PSO (CLPSO) was proposed to reduce the tendency to premature convergence adopting an update velocity rule in which the best attractor was selected with a random rule among all the particles e.g. with a tournament probabilistic selection scheme [43].

In the same year, the TRIBE algorithm was released. This algorithm is based to create a number of local neighbourhoods (the so called tribes) in which the particles shared together the best cognitive position (informant particles). The various tribes have at least one particle which is also informant of another different neighbourhood to allow information transmission among the tribes. The tribes can also be dynamically changed as stated in [70]. It is also characterized by a high self-adaptability because, according to the fitness of the particles, each tribe has a leader which perform as local attractor and through a comparison with other tribes they can also be eliminated and substituted by new ones [43] or they can be subjected to some

mutation operators to enhance the exploration capabilities [70].
When the aim is to locate and track al the local minimum of a multimodal problem, the niching PSO algorithm was proposed adopting some specific operators for this aim. In 2006 before, later in 2007 and, finally, in 2011 the Standard PSO (SPSO) tried to propose a unique baseline of the PSO algorithm for all the future implementations collecting together the so far theoretical developments in the PSO research fields.

Since the box-type search space define side constraints for each design variable ($x_j^{(l)} \leq x_j \leq x_j^{(u)}$), in the 2008 [43], the concept of opposite point give rise to the so called Generalized Opposition-based PSO (GPSO). Considering the $j$-th design variable, the opposite point of $x_j$ is defined as

$$x'_j = x_j^{(l)} + x_j^{(u)} - x_j. \tag{3.2.17}$$

In the GPSO, two different swarm are initialized: the first one is randomly selected and the second one adopt the above definition of the opposite point with a randomness coefficient $r = rand[0, 1]$, in symbols

$${}^{(0)}x'_{ij} = r\left({}^{(0)}x_{ij}^{(l)} + {}^{(0)}x_{ij}^{(u)}\right) - {}^{(0)}x_{ij}. \tag{3.2.18}$$

After the evaluation of the two swarms, the first $N$ best particles are selected to become the actual swarm of the PSO algorithm. Afterwards, at each iteration some randomness are introduced, e.g. a sort of mutation operator, to increase the exploration capability of the algorithm.

In 2012 Kar et al [32], proposed a new variant called Craziness PSO (CRPSO), introducing a craziness operator which simulate a mutation operator in the GA. In order to enhance the diversity in the population and avoid premature convergence, the velocity update rule is changed to take into account that in a real swarm the individual suddenly can change their direction in a random way. The two parameter $\boldsymbol{r}_{1i}, \boldsymbol{r}_{2i}$ of (3.1.1) are assumed to be independent, but [32] they assumed to take a unique random term $r_1 = rand[0, 1]$. To balance the various term a second random term $r_2$ was also introduced whilst a third term $r_3$ was introduce to consider a possibly random reversal of the previous velocity not related to an inertia term, with $r_2, r_3 = rand[0, 1]$,

$${}^{(k+1)}\mathbf{v}_i = r_2 \text{sign}(r_3)\, {}^{k}\mathbf{v}_i + (1-r_2)c_1 r_1 * \left[{}^{k}\boldsymbol{x}_i^{Pb} - {}^{k}\boldsymbol{x}_i\right] + (1-r_2)c_2(1-r_1) * \left[{}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i\right], \tag{3.2.19}$$

where

$$\text{sign}(r_3) = \begin{cases} -1 & r_3 \leq 0.05, \\ 1 & r_3 > 0.05. \end{cases} \tag{3.2.20}$$

After that, the velocity is further subjected to the craziness operator, that suddenly change its value with a rare craziness probability $P_{cr}$ (similarly as a mutation operator)

$${}^{(k+1)}\mathbf{v}_i = {}^{(k+1)}\mathbf{v}_i + P(r_4) \cdot \text{sign}(r_4) \cdot \mathbf{v}_i^{\text{craziness}}, \tag{3.2.21}$$

where each component $v_{ij}^{craziness}$ is randomly drawn by the velocity admissible interval $[-v_j^{max}, +v_j^{max}]$, whereas the craziness factor $P(r_4)$ is given by

$$P(r_4) = \begin{cases} 1 & r_4 \leq P_{cr}, \\ 0 & r_4 > P_{cr}, \end{cases} \tag{3.2.22}$$

and

$$\text{sign}(r_4) = \begin{cases} -1 & r_4 \geq 0.5, \\ 1 & r_4 < 0.5, \end{cases} \tag{3.2.23}$$

with $r_4 = rand[0, 1]$. As one can see, in this way, the reversal of the velocity vector is a rare event, therefore its activation is related to a very low probability ($r_3 \leq 0.05$), whereas it is suggested to adopt $P_{cr} \leq 0.3$ in order to avoid excessive oscillations when the algorithm almost reaches the convergence to the optimum. The term $\text{sign}(r_4)$ is used to consider an equal probability of reversal of the craziness velocity term [32].

Some PSO variants were also successfully adopted to deal with multi-objective optimization problems. As stated in [43], the simplest approach is the weighted aggregation method which allows to transform the problem in a single objective optimization problem. This method was already mentioned at the end of the Section 2.2 of the present Thesis. In the PSO field, early applications appeared in 2002 when the conventional weighted aggregation implemented a PSO approach with fixed weights during all the iterations. Later also dynamic approaches were proposed. Another possible approach to deal with this kind of problem with PSO is to consider a number of swarms, which work in parallel, equal to the number of the OFs to optimize. This second approach is also known as Vector Evaluated PSO (VEPSO). Each swarm is related to a specific OF but the information can be exchanged among them through certain migration schemes or neighbourhood connections as explained in [43]. In [52], the VEPSO is applied to solve a structural multi-objective optimization problem related to the optimal design of composite structures.

### 3.2.1 Structural applications and main Hybridizations

Several researchers have applied PSO algorithms to solve various types of structural optimization applications with continuous or discrete design variables, mainly for truss problems (Perez and Behdinan [57]; Li et al. [39]; Hasançebi et al. [29]; among others) and composite structures (Omkar et al. [52]; Bloomfield et al. [4]). Kaveh and Talatahari [33] combined PSO with Ant Colony Optimization (ACO) and Harmony Search (HS) to obtain a hybrid scheme which has been implemented for the optimization of truss structures. Plevris and Papadrakakis [58] combined PSO with an efficient mathematical programming method (Sequential Quadratic Programming SQP) to improve the local convergence rate of PSO via gradient-based SQP and applied this hybrid scheme to optimize typical trusses. Rao and Sivasubramanian [63] presented a computational system for the active vibration control of seismically

excited buildings by combining a multi-objective PSO algorithm with a fuzzy logic controller. Ge et al. [25] combined PSO with dynamic recurrent neural networks to perform speed control for ultrasonic motors. Begambre and Laier [1] proposed a hybrid PSO that has been combined with a Simplex algorithm to deal with structural damage identification problems. Seyedpoor et al. [72], implemented PSO for the optimum shape design of arch dams under earthquake loading using a fuzzy inference system and wavelet neural networks for the reduction of enormous computational cost. Under this perspective, Gholizadeh and Salajegheh [26] performed optimal design of steel frames subjected to earthquake loading by swarm intelligence and advanced neural network metamodels. Similarly, Praveen and Duvigneau [61] proposed the combination of PSO with radial basis function approximations to solve demanding aerodynamic shape design problems. Furthermore, in order to reduce the computational cost Kalivarapu et al. [31], presented the application of PSO in a parallel computing environment, in which digital pheromones have been used to coordinate swarms within the explored design spaces.

As one can find in [71], in the last decades' many *hybrid algorithms* were formulated in order to overcome the drawbacks of a single approach implementing different optimization strategies to find the optimal trade-off between exploration and exploitation, reducing computational efforts and avoiding the swarm to be entrapped into local sub-optimal solution. Both for constrained and unconstrained problems, there exist an integration with Genetic Algorithm operators named GA-PSO. These two approaches are referred to different contexts as one can check in [59]: GA is the oldest approach and is referred to a biological context implementing genetic operators (selection, crossover, mutation) while PSO is based on a social context. These two strategies can be used both sequentially, where PSO allows to speed up global exploration whereas GA is mainly used in the exploitation phase and also to guarantee the diversity of members in the exploration phase, or using them in parallel. Other hybrid approaches integrate Differential Evolution Algorithm, DEA (by Storn and Price, 1997) with PSO. They are also known as SDEA (Swarm Differential Evolution Algorithm) or DEPSO. In this latter, at the origins, PSO and DEAs operators sequentially work alternating at odd and even iterations [69]. In order to solve multi-objective problems an integration between PSO and Simulated Annealing (SA). Some approaches focused on implementing an adaptive updating of memory of particles' best solutions. Further hybridization was performed with Ant Colony Optimization (ACO) in order to find optimal solutions for highly non-convex problems. There are some other approaches based on Cuckoo Search (CS) which was inspired by behaviour of cuckoos integrated with Levy flight nature of birds. In these CSPSO approaches, cuckoos which reached a good solution communicate it to other members and then local exploitation of PSO is used. Later, another effort was done integrating CSPSO with DE. Always inspired by Nature, researchers proposed Artificial Bee Colony (ABC) in parallel with PSO allowing exchanging information between swarm and bees and then many other developments. There was also an integration between PSO and other social metaheuristic approaches like Artificial Immune System (AIS), Bat Algorithm (BA), Firefly Algorithm (FA), Glow Worm Swarm Optimization (GSO). As explained in [71], using algorithms in parallel allows

to improve each search mechanism thanks to information exchange between them. Mota et al. 2018 [21] implement a hybrid PSO with Iterated Local Search (ILS) operator which is based on a deterministic hill climbing phase to improve local search around current gbest.

## 3.3 State of the art in Constraint handling

As already mentioned before, the optimization problems can be classified in two main categories: if they have any external constraints which reduce the feasible search space, they are denoted as constrained problems, otherwise they are called unconstrained optimization problems. Regardless of the type of the problem, in order to avoid unbounded search which could potentially lead to an infinite number of iteration of any algorithm if an actual optimal solution does not exist, the side constraints are usually set defining an enclosed box-type search space [64]. The side constraints pose to each design variable an upper bound and a lower bound thus the search is done inside this interval. According to [70], there are at least eight strategies to deal with a particle that, after the position update given by (3.1.2), it is located outside the box-search space, which are summarized in Figure 4.9. It worth noting that in the Figure they are all depicted together to produce a compact representation, but usually only one technique is adopted for a specific implementation. In the *absorption method*, the side constraints act as an ideal wall which can absorb whichever possible impact of a particle which ends its trajectory on the side constraint. In the *reflection method* it always acts as an ideal wall perfectly smooth which is able to reflect whichever impact. The part of the trajectory which lies outside the box constraints is therefore reflected inside the box search space. In the *velocity scaling approach* when the final position is detected to lie outside the box search space, the modulus of the velocity vector is progressively reduced for example with a halve method (bisection method) or adopting certain reducing laws e.g. a hyperbolic scaling law as proposed in [70]. The *infinity method* acts similarly to a death penalty method which is discussed in the following. The most used method for its simplicity and rapid implementation, which is also adopted in this Thesis, is the *nearest method* which is mathematically described by the relations in (3.1.6). In practice, after computing the final position of the particle given by (3.1.2), this method operates on each single design variable in turn. The approach remaps the single design variable $x_j$ to the nearest boundary of the admissible interval $\left[x_j^{(l)}, x_j^{(u)}\right]$ if its value lie outside of this interval. Another trivial approach is the *random method* which randomly reinitializes the particle when they lie outside the admissible box-search space. This latter approach could also be a good approach to enhance the exploration because of the new random generation of the unfeasible particle. As stated in [70], the random method can also be applied only to that design variables which violate their admissible search interval. In the *simply periodic method* prosed in [70] foresees to ideally consider the OF on the box search space as periodically repeated all around the actual search space. In this way the particles seem unbounded in the design space but, knowing the size of the initial search space

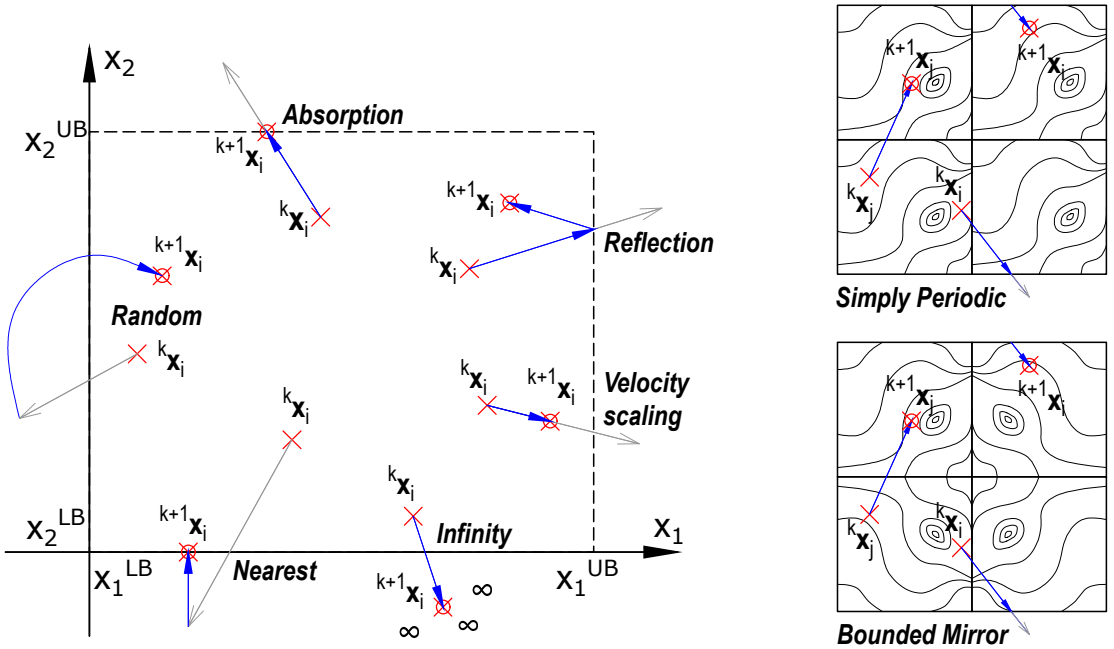



**Figure 3.3:** Visual representation of the main box-constraint approaches (image inspired by [70])

(in practice, the period), it is always possible to reconduct the position to a new position inside the original search space. A similar approach is the *bounded mirror method* where the OF is periodically repeated but this time is also mirrored. This latter acts as a combination of periodic and reflection method in which, this time, the search space is only doubled respect to its original size, as stated in [70]. Usually, also the velocities are modified when the particles violate the side constraints. The usual approach is to pose them to zero (for example in the absorption or in the nearest method) but it can also be reversed (for example in the reflection method) or even randomly changed [70].

In order to solve constrained optimization problems with PSO, several numerical techniques were incorporated for handling various types of constraints. It should be stressed that the selection of an appropriate technique for solving constrained optimization problems is a very important step because the performance of the optimizers strongly depends on the underlying mechanism for handling constraints. Motivated by this fact, various methodologies have been proposed in recent years and in this Thesis a new one is proposed. These methods have been classified by different authors into certain categories (see for instance the state-of-the-art review by Coello [10], Koziel and Michalewicz [36]; Michalewicz and Fogel [47]):

- penalty function-based methods,
- methods based on special operators and representations,
- methods based on repair algorithms,
- methods based on the separation between OF and constraints,

- hybrid methods.

One of the most critical issues when searching multi-constrained non-convex design spaces is the preservation of the population diversity. The brutal elimination of the unfeasible solutions during the evolutionary search jeopardizes a complete exploration of the feasible domain. Therefore, it is crucial to maintain diversity in the population in order to keep track of the solutions inside and outside the feasible region [10, 46]. Moreover, it has been verified that several traditional penalty-based approaches may not be adequate to deal with highly complex search spaces, especially for problems in which several constraints are active in the optimum [68]. In these circumstances, unfeasible individuals may have very important information, thus, their role can be significant when looking for the global optimal solution.

An example of a very simple constraint handling approach in PSO consists in exploiting the randomness in velocity expression (3.1.1) and recalculate it until the new position of a particle becomes feasible. This simple approach is really time-consuming in particular for problems with a little feasible region. The penalty function approach allows to transform constrained problems into unconstrained ones [30, 41]. Since its simplicity, the death penalty was the most widespread at the beginning. It introduced a strong penalty to unfeasible positions in order to consider only feasible ones. Later some authors proposed an adaptive penalty in order to evaluate the degree of violation of the unfeasible points. In fact, the optimum solution is often situated at the boundary between feasible and unfeasible region. Due to this fact, the degree of violation represents an extremely useful information in order to conduct the search along the boundary. Thanks to this latter information it is possible to implement a repairing operator which redirects the unfeasible point to the feasible region. It is important to set the velocity of redirected particles to zero in order to improve local boundary search, as illustrated in [59]. As one can check also in [30], there also exist some approaches based on searching for feasibility operators. In Kohler et al. 2016 [41], a new variant called PSO+ based on preserving of feasibility is presented. Some authors such as [41], described the constraint boundary with a wall metaphor and defined for it three different possible impact behaviour, which are also depicted in Figure 3.4. When the potential final position of a particle lies in the unfeasible region it implies that its trajectory ideally crosses the constraints function in the classic Newtonian PSO version. If the constraint acts as an ideal absorbing wall, it is able to stop the particle on its impact area. In this sense, the constraint represents a wall made of an ideal material which is able to adsorb whichever energy of any impact and stopping the particle on its surface. Instead, in the reflection wall concept, the constraint acts as an ideal perfectly reflecting wall. In fact, when a particle would cross the constraint surface, it will impact with this surface and the particle is rebounded behind without loss of energy. This means that the final position reached after the impact would be the reversed part of the uncovered trajectory in the unfeasible region. In general, the constraint usually acts as an invisible wall: it means that it does not influence the trajectory of the particle in any way. This implies that the preservation of the feasibility has to be treated in another way with specific operators. Unfortunately, this wall metaphor can be simply applied to the side constraints of each design variable, but it is much more difficult to implement when one is dealing with general non-linear constraints

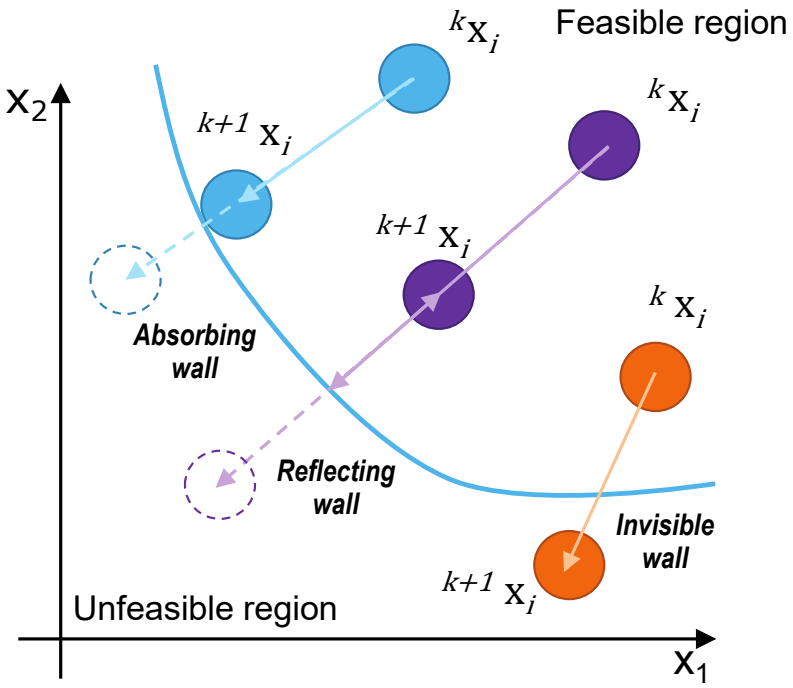



**Figure 3.4:** Graphical representation of the constraint wall theory exposed in [41].

functions defined inside the design space. In Kohler et al. 2016 [41], a new variant called PSO+ based on preserving of feasibility is presented. In this approach, an arithmetical crossover operator is used to computationally improve the generation of feasible initial population. Then the search for the new positions is conducted with the Footholds concept. Footholds are some fixed random generated unfeasible points which do not belong to the swarm but they can affect the search direction likewise e.g. the gbest point. The unfeasible particles are then repaired with the crossover operator in order to obtain again a feasible point. For further information, Jordehi 2014 [30] has made a review of the general constraint handling methods in heuristic algorithms mainly focusing on pro and cons of the methodologies applied in PSO. Herein, it illustrated main characteristics about static, dynamic and death penalties approaches, separatist mechanism such as Deb's approach, MO-based separatist mechanism, Co-evolutionary-based separatist mechanism, stochastic ranking $\alpha$-Constrained PSO which define a satisfaction level, $\varepsilon$-Constrained PSO which introduce a relaxation factor in Deb's approach and other constraint mechanisms such as Del Valle's approach and general hybrid approaches.

### 3.3.1 Penalty function-based methods

One of the most used indirect technique to solve constrained optimization problem is represented by the penalty function-based methods. These methods allow to transform the constrained optimization problem in an equivalent unconstrained version. As a matter of fact, they tend to modify the actual OF in order to take into account of the presence of the constraints which divide the box-search space in the feasible and the unfeasible regions. The actual OF $f(\boldsymbol{x})$ of the original optimization problem stated in (2.1.2) is transformed in a new OF $\phi(\boldsymbol{x})$ by adding, in the most general way, a certain penalty function $P(\boldsymbol{x})$. This penalty function $P(\boldsymbol{x})$, in general, is equal to the sum of a penalty function related to the inequality constraints $G(\boldsymbol{x})$ and a term related to the penalty of the equality constraints $H(\boldsymbol{x})$. These two terms are, in general, respectively weighted by two positive penalty factors $c$ and $k$. In symbols [30]

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + P(\boldsymbol{x}) = f(\boldsymbol{x}) + \left[ \sum_{j=1}^{n_q} c_j G_j(\boldsymbol{x}) + \sum_{i=1}^{n_r} k_i H_i(\boldsymbol{x}) \right], \tag{3.3.1}$$

where $n_q$ is the total number of inequality constraints, and $n_r$ is the total number of equality constraints.

As stated in [64], there exist essentially two types of penalty functions. The *interior penalty* functions act in the feasible region (this explains their appellation) augmenting the value of the OF where the design space approaches the constraints. When a particle is in the feasible region, the penalty function tends to be zero, whereas if the particle comes closer to the constraint, the penalty function increases its value till blowing up to infinite when it is critically satisfied [64]. This means that the constraint acts as an ideal barrier and for this reason this type of penalty is also called barrier method [30, 64], reminding the above mentioned wall metaphor [41]. Two examples of most used interior penalty function forms are

$$G_j(\boldsymbol{x}) = -\frac{1}{g_j(\boldsymbol{x})}, \qquad G_j(\boldsymbol{x}) = -\log\left[-g_j(\boldsymbol{x})\right]. \tag{3.3.2}$$

With these interior models there are many drawbacks because, at first, they necessary require to work with initially feasible points, which can lead to complications in the initialization of the swarm or a higher computational effort. Furthermore, it is complicated to establish a prior suitable value of the penalty factors, therefore, in [64], a dynamic approach is proposed in which the penalty factors start with a great value and, during the iterations, they tend to zero in order to compensate a probable blow up of the penalty term.

On the contrary, the second type of penalty functions are called *exterior penalty*. These functions act a modification of the OF in the unfeasible region and maintain the actual OF in the feasible region. Because this feature, it is able to take into account the degree of violation of the constraints. Therefore exterior penalty functions take, in general, the following form for the inequality constraints

$$G_j(\boldsymbol{x}) = \max\left\{0, g_j(\boldsymbol{x})\right\}^{\beta}, \tag{3.3.3}$$

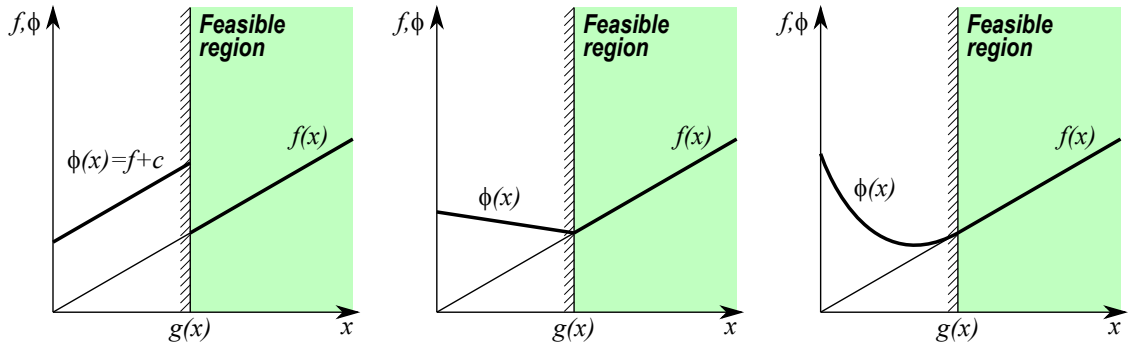



**Figure 3.5:** Graphical representation of the exterior penalty exponent influence on a unidimensional example. On the left, $\beta = 0$; in the center, $0 < \beta \leq 1$; on the right, $\beta = 2$; image inspired by [64].

where $\beta$ is a positive constant and

$$\max\{0, g_j(\boldsymbol{x})\} = \begin{cases} g_j(\boldsymbol{x}), & \text{if } g_j(\boldsymbol{x}) > 0 \text{ (constraint is violated)}, \\ 0, & \text{if } g_j(\boldsymbol{x}) \leq 0 \text{ (constraint is satisfied)}, \end{cases}$$

whilst, for the equality constraints, the penalty function terms take the following form

$$H_i(\boldsymbol{x}) = |h_i(\boldsymbol{x})|^{\gamma} \tag{3.3.4}$$

where $\gamma$ is a positive constant. As stated in [30], $\beta$ and $\gamma$ are usually set to 1 or 2 in order to give a further weight to the degree of violation, whereas the penalty factors $c_j$ and $k_i$ have to be carefully chosen. As a matter of fact, in [64] it is highlithed that if the OF have a minimum point in the unfeasible region, if the penalty are not big enough, the algorithm can stall and spend a lot exploration in the unfeasible region without finding a solution which respect the constraints. Instead, if the penalty factors are too large the unfeasible region risk to not be sufficiently explored, as affirmed in [30]. Because of the consideration of the degree of violation without modifying the OF inside the feasible region, the external penalty approaches are the most commonly adopted when the constraints are handled with a penalty-based approach. In Figure 3.5 the influence of $\beta$ is represented on a unidimensional example, inspired by [64]. In fact, if it is set to zero, the only penalty added is the penalty factor leading to a piecewise function with a jump discontinuity, whereas if the value is $0 < \beta \leq 1$ the function is continuous but with a corner point, and if it is set to $\beta = 2$ the function is still continuous but, at a very low degree of violation it tend to follow the original OF whereas, when the violation increase, the penalty factor increase with power two. Similar considerations are done for the $\gamma$ exponent.

In literature [64, 79], there also exist some modifications of the classical interior penalty functions which are called extended interior penalty functions. These variants are a combination of the interior and the exterior penalty function methods. In these approaches, the condition of blowing up to infinite when the constraints are critically satisfied is removed and the penalty start to increase the actual OF in the feasible region and still continues with a certain slope (linear, quadratic) also in

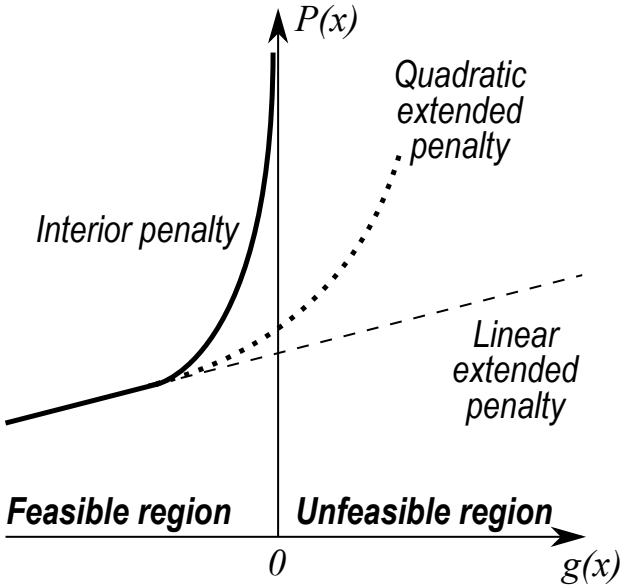



**Figure 3.6:** Graphical representation of the extended penalty approach; image inspired by [79].

the unfeasible region. In Figure 3.6 an example of the extended penalty approach is represented.

In [30], a review of the most used constraint handling approaches adopted in PSO is presented. The *static penalty-based* techniques are one of the most popular tools in structural optimization, as stated in [29]. They are called in that way because the penalty functions do not change during the proceeding of the iterations of the algorithm. The most popular form of static penalty function is [12, 30]

$$P_s(\boldsymbol{x}) = w_1 P_{NVC}(\boldsymbol{x}) + w_2 P_{SVC}(\boldsymbol{x}), \tag{3.3.5}$$

in which $P_{NVC}$ is the number constraints that are violated by the particle $\boldsymbol{x}$ and $P_{SVC}$ is the sum of the constraint violations

$$P_{SVC}(\boldsymbol{x}) = \sum_{p=1}^{n_p} \max\{0, g_p(\boldsymbol{x})\}, \tag{3.3.6}$$

where $n_p$ is the total number of constraints, whereas $w_1$ and $w_2$ are the static weight parameters of the penalty scheme. Unfortunately, in literature, there is no concordance in a proper suggested choice of the penalty parameters, also because they are problem dependent and, therefore, they have to be carefully tuned for the specific problem with a trial and error procedure. They are usually assumed as integer between 1 and 999 as stated in [12] and, for example, [30] stated that in an application the authors used the value 20 without any apparent reason or, as well as, Parsopoulos and Vrahatis [56] which adopted $w_1 = w_2 = 100$.
A three-dimensional graphical representation of the static penalty is given in Figure 3.7 (d). As one can see, the static penalty acts an uplift of the actual OF only in the unfeasible region, in which the height of the uplift depends on how much the

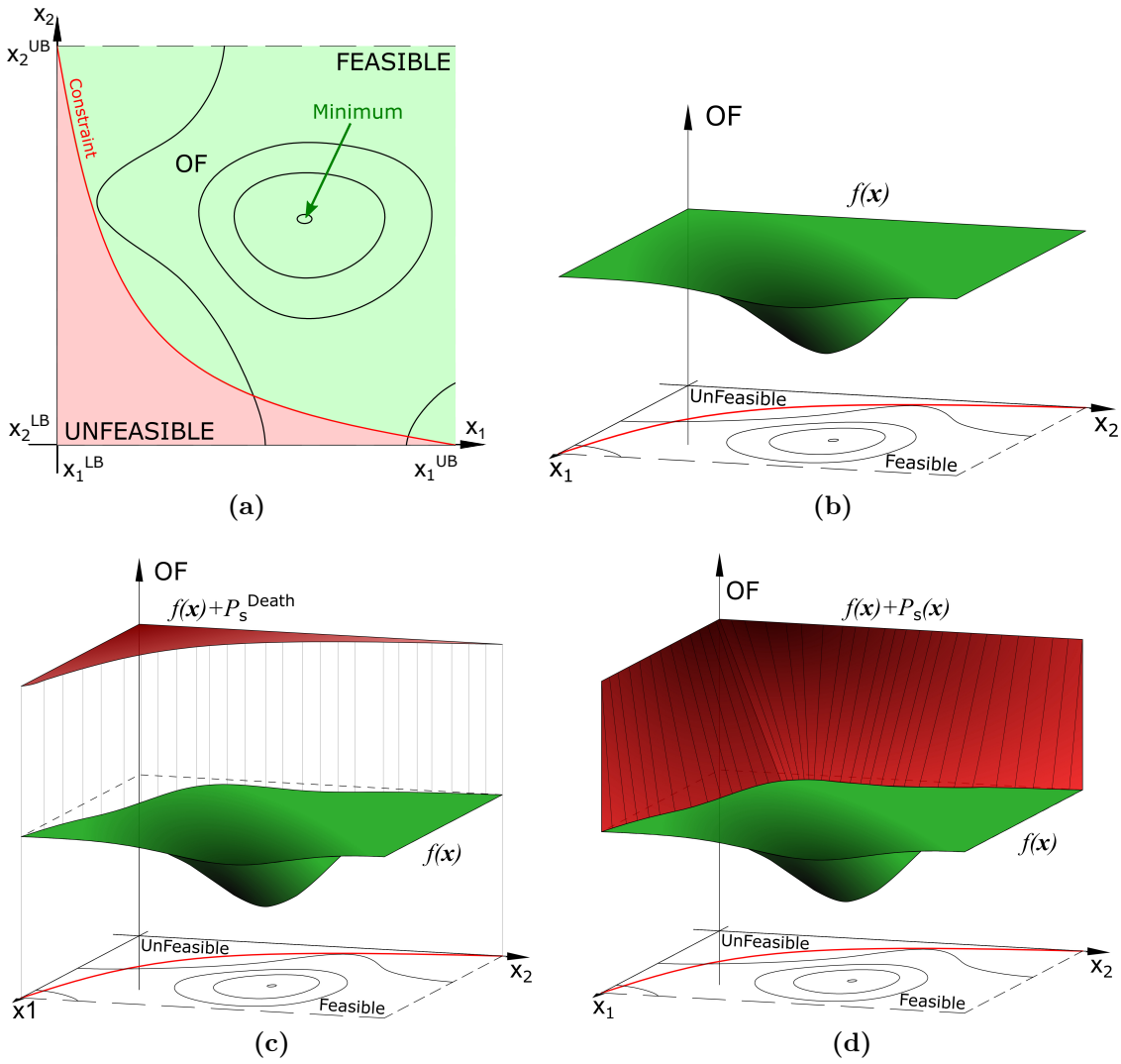



**Figure 3.7:** Graphical representation of the *penalty-function approach* on a bi-dimensional example. (a) Planar view of the OF $f(\boldsymbol{x})$ represented as contour black solid lines in the design variables' space. The constraint, depicted as the red solid line, divide the feasible region from the unfeasible one. The box search space boundaries are drawn as dashed lines: $x_i^{(l)}$ is the design variable lower bound and $x_i^{(u)}$ is the design variable upper bound, with $i = 1, 2$; (b) Three-dimensional representation of the OF surface on the entire box search space; (c) Death penalty approach graphical representation; (d) Static penalty approach taking into account the degree of violation of the constraint.

constraint is violated.

The extreme case of the static penalty approach is represented by the so called *death penalty*. The main advantage is that this approach avoids the choice of the penalty parameters because they are set to infinite [30]. This fact lead to have a piecewise OF with a jump discontinuity across the constraints, as depicted in Figure 3.7 (c). From the numerical point of view, to simulate the infinite value, the penalty parameters are set to a value greater several order of magnitude respect to the actual OF.

In general, the penalty function strongly modifies the performance of the algorithm when it deal with constrained optimization problems. To improve the quality of the results, some *dynamic penalty functions* were proposed to increase the algorithm's performance. They are also called non-stationary penalty function in [64] because the penalty parameters dynamically change with the iteration number $k$.

To this end, the equation (3.3.1) is modified as follows

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + {}^{k}hP_d(\boldsymbol{x}) \tag{3.3.7}$$

in which ${}^{k}h$ is a dynamic penalty whose numerical value is suggested to be [64, 30]

$${}^{k}h = (ck)^{\alpha} \tag{3.3.8}$$

where $c$ is a penalty coefficent and $\alpha$ is an exponent which is suggest to be equal to 2 in [64]. The the dynamic penalty function $P_d(\boldsymbol{x})$ is given by [30]

$$P_d(\boldsymbol{x}) = \sum_{i=1}^{n_r} H_i(\boldsymbol{x}) + \sum_{j=1}^{n_q} \left[G_j(\boldsymbol{x})\right]^{\beta}, \tag{3.3.9}$$

where $n_q$ is the total number of inequality constraints, and $n_r$ is the total number of equality constraints, with

$$H_i(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \varepsilon \leq H_i(\boldsymbol{x}) \leq \varepsilon, \\ H_i(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{3.3.10}$$

and

$$G_j(\boldsymbol{x}) = \max\left\{0, g_j(\boldsymbol{x})\right\}. \tag{3.3.11}$$

As proposed in [30], transforming all the constraints in inequality constraints, a multi-stage dynamic penalty approach for PSO was also proposed. The modified OF had the same the form of (3.3.7), but the dynamic penalty factor was assumed as [64, 30]

$${}^{k}h = \sqrt{k}, \tag{3.3.12}$$

and the penalty function assumed the following form

$$P_d(\boldsymbol{x}) = \sum_{p=1}^{n_p} \theta\left(q_p(\boldsymbol{x})\right) q_p(\boldsymbol{x}) e^{\gamma(q_p(\boldsymbol{x}))}, \tag{3.3.13}$$

with $q_p(\boldsymbol{x}) = \max\left\{0, g_p(\boldsymbol{x})\right\}$ where typical assignments for the penalty parameters are (see for instance [55])

$$\theta\left(q_p(\boldsymbol{x})\right) = \begin{cases} 10 & \text{if } q_p(\boldsymbol{x}) \leq 0.001, \\ 20 & \text{if } 0.001 < q_p(\boldsymbol{x}) \leq 0.100, \\ 100 & \text{if } 0.100 < q_p(\boldsymbol{x}) \leq 1.000, \\ 300 & \text{otherwise.} \end{cases} \tag{3.3.14}$$

$$\gamma\left(q_p(\boldsymbol{x})\right) = \begin{cases} 1 & \text{if } q_p(\boldsymbol{x}) \leq 1, \\ 2 & \text{otherwise.} \end{cases} \tag{3.3.15}$$

It is evident that dynamic penalty methods require a larger number of control parameters in comparison to the static one, however, it was numerically demonstrated to give better results respect the static approaches [64].

Since the presented dynamic method also take into account the degree of violation, they could also be classified as adaptive penalty methods. In reality, in the adaptive penalty methods, the penalty parameter is defined at each $k$-th generation for each $j$-th constraint taking into account both the OF value of the particle $\boldsymbol{x}$ and also the global degree of violation, e.g. [7]

$$^{k}h_j = |f(\boldsymbol{x})| \frac{q_j(\boldsymbol{x})}{\sum_{p=1}^{n_p} \left[q_p(\boldsymbol{x})\right]^2}, \tag{3.3.16}$$

with $q_j(\boldsymbol{x}) = \max\{0, g_j(\boldsymbol{x})\}$. For furthermore readings about adaptive penalty schemes one can refer to [7], in which this latter technique was adopted in CRPSO for solving structural optimization problems.

# Chapter 4

# A new non-penalty Machine Learning constraint handling approach

In the following, after a brief excursus on Artificial Intelligence, a review on the support vector machine (SVM) technique is presented and, after that, the new constraint handling non-penalty proposed approach is reported in detail. The main advantage of adopting a new non-penalty based constraint handling approach is related to the generality of the classification machine learning algorithm employed. As a matter of fact, the SVM depends only on the inner product of the data and it is able to generate a predictive model. This latter represents substantially the boundary between the feasible positions of the swarm from the unfeasible ones. This predictive model is more adaptive than a typical penalty function approach because works fine both with discontinuous and non-linear constraints. In the next chapter, some practical applications of this new PSO-SVM proposed variant are presented.

## 4.1 Artificial Intelligence: brief overview and main scopes

Artificial Intelligence (AI) is a very large research field which is still under development and it represents the present but also the future of the research. The purpose of this section is to make a brief excursus of AI and understand where the metaheuristic algorithms are located in this vast research field. Especially related to the computer science field, in [20] AI is defined as the process to simulate the human brain reasoning on a digital computer:

> "*Through research of intelligent systems we can try to understand how the human brain works and then model or simulate it on the computer. [...] AI has as its constant goal the creation of intelligent agents for as many different tasks as possible.*".

According to the above definition, AI aims to produce systems that are able to reason, solve problems and take decisions an intelligent manner likewise a human

brain would have done. The strongest point of human intelligence still remains the great adaptability to many situations exploiting a process called learning, therefore the machine learning represents one of the most important branches of AI [20]. From the above definition, it is even possible to state that it is not possible to realize a perfect system which is able to deal with any type of problem. For this reason, AI adopts different agents whose features are different according to the target they aim. An agent is defined as a program (in computer science) or, in more general terms, a system which is able to process some input data and give an output. In mathematical terms, it is substantially a map function of the inputs to the outputs. The agents live in a certain space which is called environment and it can sound its state through sensors which give back some feedback. The agents can also interact with the environment changing its current state through their actions or decisions. As stated in [20], these agents are classified in different categories in function of the type of intelligence. If they react in function of the input and also from a past memory, they are called *agents with memory*, whereas if they only react to the current input and do not possess any memory, they are called *reflex agents*. These latter are usefully adopted in Markov decision processes which foresee to take an optimal decision only based on the knowledge of the current state or conditions [20]. The *learning agents* are a different type of agent which can change the mapping of the input to the output in function of new information given by a training phase e.g. from known examples or exploiting the feedbacks given back from the sensors. When intelligent behaviour is exhibited from some interactions of the entire group of agents, they are called *distributed agents*. Therefore, adopting the AI language, the PSO is composed of learning distributed agents with memory.

From the historical point of view, the bases of AI were set in the 1930s with Gödel and Turing which founded logic and the theoretical computer science [20]. Someone else set the born of AI in 1943, when McCulloch and Pitts formalized the first model of neural network with two neurons, describing the simplified working mechanism of a single neuron from the neuroscience in a mathematical formulation. The name "Artificial Intelligence" was firstly adopted by McCarthy in 1956 [20]. As reported in [43], in the early period, also the meta-heuristic algorithm were associated to AI, because they also try to mimic the intelligent behaviour of species of animals in the natural environment with the Darwinian scheme (EAs). Later, starting from the 1960s, the two disciplines started to take different directions, however they still remain in contact even nowadays. As a matter of fact, nowadays, AI and the meta-heuristic techniques are sometimes considered two different fields whereas sometimes AI is considered the general field which embeds also a branch called *computational intelligence* which comprises EAs, Swarm intelligent algorithm and, in general, all the meta-heuristic approaches. As a matter of fact, as reported in [62]

> "*the computational intelligence [...] is defined as a branch of artificial intelligence dealing with the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments*."

Nowadays, AI field is a multidisciplinary field and its scopes and methods can be classified in different ways relating to the branch of Science these techniques applied and/or adapted. In more general terms, the part of AI which aims to develop

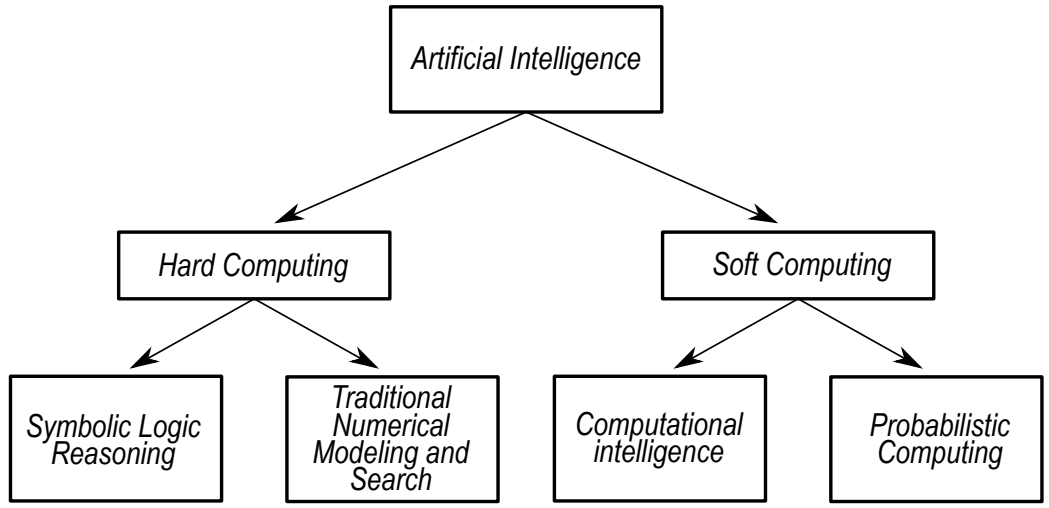



**Figure 4.1:** Scheme of AI applied for problem-solving computing techniques (machine intelligence), image inspired by [22].

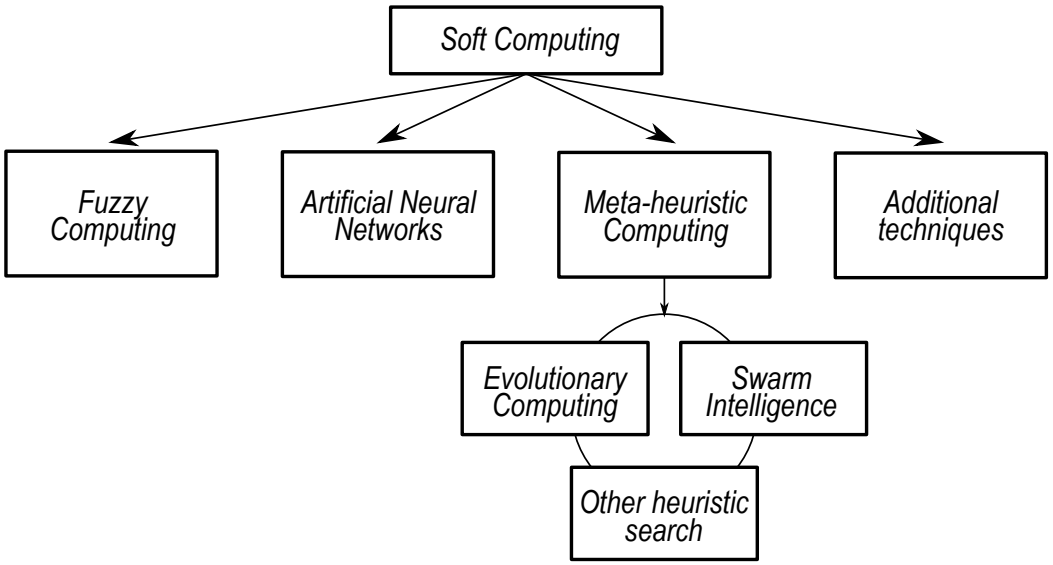



**Figure 4.2:** Scopes of Soft Computing sub-field in order to locate the meta-heuristic approaches in the vast field of AI; image inspired by [22].

intelligent algorithms for the problem-solving is called machine intelligence or simply AI. This vast branch consists of two different sub-fields. The first one is the so called *hard computing* which is related to solve problems in a traditional way, therefore adopting analytical resolutions, mathematical programming techniques and sequential approaches to reach an exact solution [22]. These techniques are adopted with precise analytical modelled problems and adopt exact resolution methods. The second sub-field is the so called *soft computing*. According to [22], this term was introduced for the first time by Zadeh in 1992 and it is defined as "*the union of computational intelligence paradigms and probabilistic computing*" as stated in [62].

This implies that the computational intelligence is an embedded part of the more vast field of soft computing techniques. In fact, as stated in [22], soft computing is divided into the so called approximate reasoning and the randomize search approaches. In these techniques, the problems to solve can also take into account the uncertainties, which are typical in civil engineering problems, and also of ambiguity or noise input or even verbose problem statements which are dealt with the so-called Fuzzy Logic techniques [64]. As presented in Figure 4.1, a summary scheme of the machine intelligence scopes is presented and, in Figure 4.2, the main approaches of soft computing are depicted. Thanks to these representations is now possible to better locate the PSO algorithms inside the vast world of AI. A review of the most adopted soft computing technique for modeling, simulation and optimization problems in Civil Engineering for can be found in [22].

Due to the fact that all these fields are still under research, there is no concordance in the scientific community about this classification and it isn't the unique possible, but it seems to be the best representative for the current state of the art.

One of the most current important fields in AI is the so called *machine learning* which is defined as "*the application and science of algorithms that make sense of data*" in [65]. This brief definition exposes the aim of the machine learning which is to develop algorithms to analyze data in order to identify schemes or, in general, extract apparently hidden information through an automatic learning approach. The obtained output model of the data is, in general, a predictive model which is used to make future forecast and decisions e.g. with adding new input data. There are two main types of problems in which machine learning is adopted: the first one is the classification problems in which the data have to be labelled in different classes, whereas the regression problem aims to define a predictive model to explain how continuous data are correlated [65]. The data are organized in samples and, before starting the algorithm, may be necessary to preprocess them in order to improve the performance of the machine learning algorithm e.g. adopting a normalization process and scaling of the data. In particular, a fundamental step in this phase is to separate the data in a training set and in a test or validation set with usual split proportion as 60:40 or 70:30 or even 80:20 on the entire available dataset [65]. In fact, as affirmed in [64], almost all the numerical procedures can fail because of several orders of magnitude of difference among the values of the design variables. This could happen in structural optimization problems e.g. where the length of the elements is at the order of some meters and the thickness of the steel section profiles is at the order of few millimeters. This aspect can lead to biased and distorted evaluation of objective function and numerical errors. One possible solution is to scale all the design variables making them dimensionless so that all of them can vary between 0 and 1. The same problem could happen in the evaluation of the constraints which need to be also normalized.
Afterwards, the initial phase of a machine learning algorithm is the training phase. In particular, there are three main types of learning processes. The supervised learning requires both input and output data which have to be priorly known (priorly labelled) to allow the training phase. In the unsupervised learning, the output data are unknown, therefore the data are not priorly labelled but these techniques try to

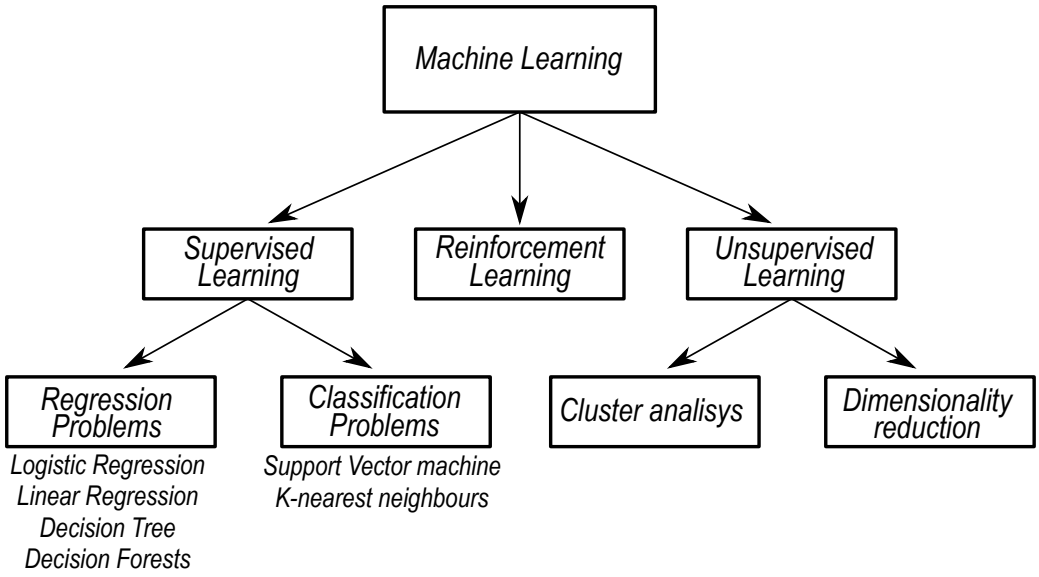



**Figure 4.3:** Representation of the machine learning main approaches: supervised learning, reinforcement learning and unsupervised learning.

find intrinsic structures inside the given dataset based on measurements of similarities among them such as in the cluster analysis. The reinforcement learning is based on the AI learning agents which has sensors which can acquire new information during the learning phase through interactions with the environment. In that sense, the reinforcement learning process can be considered a supervised learning process because the output is known after the agents received the feedback (reward signal) from the sensors [65]. After the learning phase and before performing the prediction on new unseen input data, the test data set is adopted to evaluate the quality of the predictive model e.g. through cross-validation techniques. This phase is fundamental because it allows to detect possible underfitting or overfitting problems. The underfitting is substantially when the predictive model does not represent the real structure of the data because of a lack of learning (it is said that the predictive model has a high bias). On the contrary, the overfitting problem is when the predictions perfectly fit with the test subset but it badly performs with new unseen data. In that sense, in the overfitting, the predictive model has been too much adapted to the initial data and it is not able to correctly label new unseen data. In that case, it is said that the predictive model has a high variance because it excessively fit the training data introducing unnecessary oscillations and errors in labelling new data.

As presented in Figure 4.3, the most adopted machine learning techniques are the Neural Networks (NN), which insight nowadays identify a new sub-branch called Deep Neural Networks (DNN), logistic regression, the Support Vector Machine, the decision tree and the decision forest, the k-nearest neighbour (KNN), as supervised learning methods, and the cluster analysis and dimensionality reduction, as unsupervised methods. For further insight in the machine learning field, one can refer to [3, 51, 65]

In the present Thesis, a novel non-penalty constraint handling approach tries to combine the Support Vector Machine to create predictive model in order to handle constraints in the search space of the optimization problem under study. In the following section, a brief theoretical review of the Support Vector Machine is presented.

## 4.2 Support Vector Machine: theoretical overview

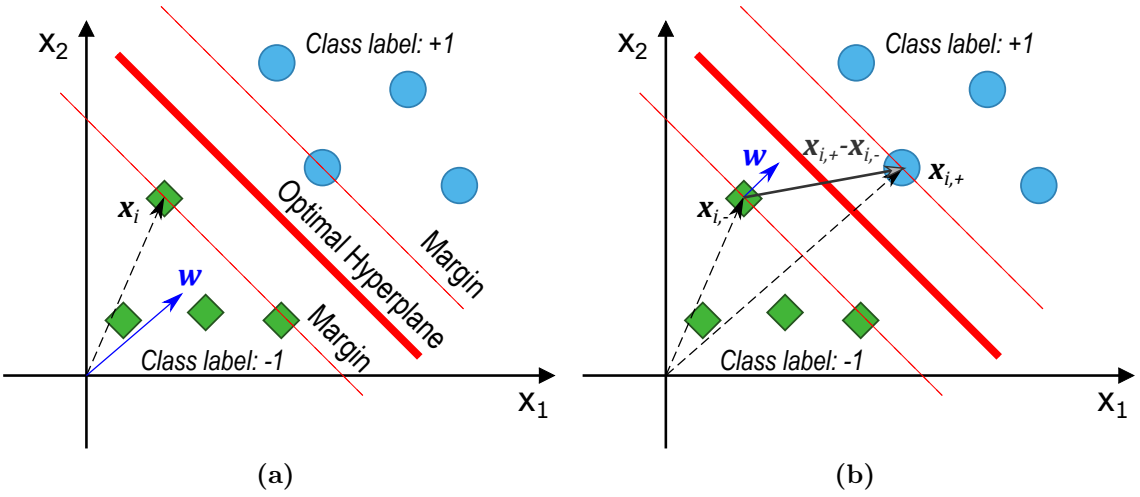



**Figure 4.4:** Support Vector Machine pragmatic graphical example in two dimensions $(x_1, x_2)$: on the left the optimal hyperplane is showed according to the rule of the maximum margin; on the right it is graphically demonstrated how to obtain the measurement of the distance between the two margins.

Support Vector Machine (SVM) is a supervised classification learning method based on statistical learning theory [80]. During the learning process, the machine "*learns from examples*" [80] contained in a known training data set which could be composed by both input and output data (supervised) or only input data (unsupervised) [3]. The result of this process is a predictive model that could map the output of other new input data. If the output is a variable from a finite set which represents the class (or category or label) of the input data, the problem is defined as a classification (or pattern recognition) problem [3, 51].

As depicted in Figure 4.4 (a), a two-class classification problem (even called binary classification [3]) is now considered in which a training data set defined as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ are the $N$ known data labelled by $y \in \{+1, -1\}$. In order to classify new input data, it is necessary to define a decision boundary and the data are defined as linearly separable [80] if exists a vector $\boldsymbol{w}$ (linear separator) and a scalar $b$ (bias) for which the decision rule takes the following form:

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1 \quad &\text{if} \quad y_i = 1, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 \quad &\text{if} \quad y_i = -1. \end{aligned}$$

where the operator " $\cdot$ " denotes the inner product. For convenience, it is possible to rewrite the two inequalities as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1. \tag{4.2.1}$$

The parameters $\boldsymbol{w}$ and $b$ may define different possible hyperplanes[1] which linearly separate data. In SVM a unique optimal hyperplane is defined according to the

[1] As stated in [64], an hyperplane in $n$-dimensional space is defined as a set of points which

principle of the maximal margin [14] which is *"the smallest distance between the decision boundary and any of the samples"* [3]. The optimal hyperplane is in the following form

$$\boldsymbol{w}_0 \cdot \boldsymbol{x} + b_0 = 0. \tag{4.2.2}$$

As depicted in Figure 4.4 (b), it is possible to give a graphical representation of the method to obtain the width $W$ of the maximal margin. Considering that $\boldsymbol{w}$ is always orthogonal to the margins and these latter have always to pass through the extreme points which are the closest points belonging to the two different classes denoted as $\boldsymbol{x}_+, \boldsymbol{x}_-$, it is possible to write

$$W = (\boldsymbol{x}_+ - \boldsymbol{x}_-) \cdot \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \tag{4.2.3}$$

where $\|\boldsymbol{w}\|$ is the norm of the vector $\boldsymbol{w}$. Recalling the (4.2.1), it becomes an equality for those points which belong to the margin. For the positive labelled sample $\boldsymbol{x}_+$ which label is $y_+ = +1$, it is possible to write that

$$(\boldsymbol{w} \cdot \boldsymbol{x}_+ + b) = 1, \quad \Rightarrow \quad \boldsymbol{w} \cdot \boldsymbol{x}_+ = 1 - b, \tag{4.2.4}$$

whilst for the negative labelled sample $\boldsymbol{x}_-$ which label is $y_- = -1$, it is possible to write that

$$-(\boldsymbol{w} \cdot \boldsymbol{x}_- + b) = 1, \quad \Rightarrow \quad -\boldsymbol{w} \cdot \boldsymbol{x}_- = 1 + b. \tag{4.2.5}$$

Therefore, considering the (4.2.3) and substituting the terms obtained in (4.2.4) and (4.2.5), the term $b$ is simplified and the resulting width is equal to

$$W = \frac{2}{\|\boldsymbol{w}\|} \tag{4.2.6}$$

According to the maximal margin principle, it is possible to demonstrate which to maximize the margin correspond to $\max 1/\|\boldsymbol{w}\|$ (neglecting the constant 2) which, for mathematical convenience, is equivalent to the following quadratic programming problem (quadratic OF subjected to linear constraints) [3, 14]: *For all* $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, *find* $\boldsymbol{w}$ *and* $b$ *such that*

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2, \\ \text{s.t.} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1. \end{aligned} \tag{4.2.7}$$

Using the Lagrange multiplier rule, it is possible to solve the quadratic optimization problem defining the Lagrangian such as

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left[ y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) - 1 \right], \tag{4.2.8}$$

coordinates respect the following linear condition $a_1x_1 + ... + a_nx_n = \boldsymbol{a}^T\boldsymbol{x} = b$. An hyperplane in $n$-dimensional space has $n-1$ dimensions. As a matter of fact, in three-dimensional space it is a plane, whilst in a bi-dimensional space it is a line. The hyperplane separates the space in two closed half-space which can be respectively defined as those set of points for which $\boldsymbol{a}^T\boldsymbol{x} \geq b$ and $\boldsymbol{a}^T\boldsymbol{x} \leq b$ [64].

where $\alpha_i$ are the not negative Lagrange multipliers. Imposing the stationariness of the (4.2.8) respect to $\boldsymbol{w}$ and $b$ one obtains

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \qquad \Rightarrow \qquad \boldsymbol{w}_0 = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i, \tag{4.2.9}$$

$$\frac{\partial L}{\partial b} = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{N} \alpha_i y_i = 0, \tag{4.2.10}$$

where (4.2.9) is called decision vector. According to [14] it is worth noting that *"the optimal hyperplane solution can be written as a linear combination of training vectors"*. Substituting the (4.2.9) in the (4.2.8) and taking into account the (4.2.10), one finally obtains the following dual problem: *Find* $\alpha_i$ *such that*

$$\begin{aligned} \max \quad & L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & \alpha_i y_i \geq 0, \qquad \forall i = 1, ..., N. \end{aligned} \tag{4.2.11}$$

It is important noting that, in the (4.2.11), the optimal hyperplane depends only on inner product of pairs of samples [80]. The optimal solution must satisfy the Karush-Kuhn-Tucker (KKT) conditions which state that "*any Lagrange multiplier and its corresponding constraint are connected by an equality*" [14]

$$\alpha_i \left[ y_i (\boldsymbol{x}_i \cdot \boldsymbol{w}_0 + b_0) - 1 \right] = 0, \quad i = 1, ..., N. \tag{4.2.12}$$

Since the (4.2.12) holds, it implies that the Lagrange multipliers $\alpha_i$ are different from zero only for a small amount of vectors $\boldsymbol{x}_{(SV)}$ which belong to the margin. These particular vectors, for which the (4.2.1) become an equality

$$y_i (\boldsymbol{x}_i \cdot \boldsymbol{w}_0 + b) - 1 = 0, \tag{4.2.13}$$

are called *support vectors* [3, 14]. The sum in the (4.2.9) is therefore restricted to the smaller set which considers only the support vectors [38]

$$\boldsymbol{w}_0 = \sum_{i \in \mathrm{SV}} \alpha_i y_i \boldsymbol{x}_i. \tag{4.2.14}$$

This is the main property of SVM called sparseness which identifies the SVM also as a Kernel Sparse Method in [15]. During the classification process "*once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained*" [3]. The result of the prediction not change if one considers either all the training data set or only the support vectors as the training data set [15].

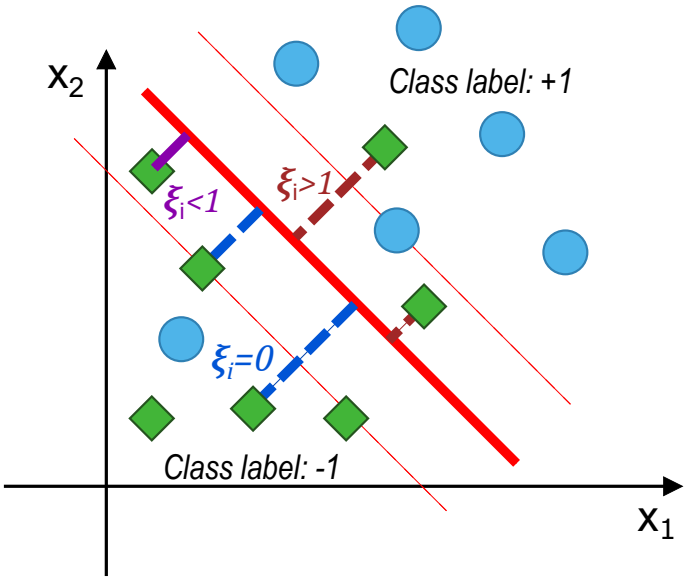



**Figure 4.5:** Support Vector Machine: soft margin concept and slack variables $\xi_i$ graphical representation in a bi-dimensional problem with non-separable data.

Non-separable training data are defined as those data which can not be linearly divided without committing a certain error [14]. One can refer to the situation depicted in Figure 4.5. According to [38], in order to define the optimal hyperplane called, in this case, soft margin hyperplane, the slack variables $\xi_i \geq 0$ are introduced, one for each datum point. These slack variables measure the classification error in terms of distance from the margin as depicted in 4.5. A slack variable is zero if the point lie inside the correct margin boundary, otherwise it takes value in the interval $(0, 1]$ if the point is inside the margin and finally its value would be greater than one if it lies on the wrong side [3, 51]. Afterwards, similar passages than the above mentioned for the linear separable data are followed. The main idea of this method is to minimize the sum of the deviations $\sum_i^N \xi_i$, besides to maximize the width of the margin. Therefore, it is possible reconducts these two aims to minimize the following functional [3]

$$C \sum_{i=1}^{N} \xi_i + \frac{1}{2}\|w\|^2, \tag{4.2.15}$$

in which the parameter $C > 0$ a penalty parameter. The statement of the minimization problem becomes the following: *For all* $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, *find* $\boldsymbol{w}$ *and* $b$ *such that*

$$\begin{aligned} \min \quad & C \sum_{i=1}^{N} \xi_i + \frac{1}{2}\|w\|^2, \\ \text{s.t.} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{4.2.16}$$

Applying the same procedure as before with the Lagrange multiplier method, in the end, one can obtain the dual formulation which is similar to linear separable case

given by the (4.2.11). The Lagrangian function of the problem (4.2.16) become [15]

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{\|w\|^2}{2} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N}\mu_i\xi_i, \tag{4.2.17}$$

where the $\alpha_i, \mu_i \geq 0$ are the Lagrange multipliers [3]. Imposing the stationariness of the Lagrangian it is possible to obtain

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} y_i\alpha_i\boldsymbol{x}_i = 0, \tag{4.2.18}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \tag{4.2.19}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i\alpha_i. \tag{4.2.20}$$

Substituting the above equations obtained by the stationariness inside the starting Lagrangian function (4.2.17), the dual OF is given by

$$L = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i\cdot\boldsymbol{x}_j \tag{4.2.21}$$

which is identical to the linear separable case [15]. The main difference is the presence of a new constraint $C - \alpha_i - \mu_i = 0$, which implies that $\alpha_i \leq C$ since $\mu_i \geq 0$ due to the Lagrange multipliers method. According to [3], if $\alpha_i < C$ the (4.2.19) implies that $\mu_i > 0$, but remembering the KKT condition which state $\mu_i\xi_i = 0$, it require that $\xi_i = 0$. On the contrary, if $\mu_i = 0$ it implies that $\xi_i \neq 0$ and therefore $\alpha_i = C$ [15]. The remaining KKT complementary conditions [15] are

$$\begin{aligned} &\alpha_i\left[y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) - 1 + \xi_i\right] = 0, \\ &\xi_i(\alpha_i - C) = 0, \end{aligned} \tag{4.2.22}$$

which implies that $\xi_i$ are different from zero only if $\alpha_i = C$ to respect the conditions, which are those points which lie inside the margin and can be correctly labelled if $\xi_i \leq 1$ or misclassified if $\xi_i > 1$ [3]. The statement of the dual problem becomes: *Find $\alpha_i$ such that*

$$\begin{aligned} \max \quad & L = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\boldsymbol{x}_i\cdot\boldsymbol{x}_j\right) \\ \text{s.t.} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad \forall i = 1, ..., N \end{aligned} \tag{4.2.23}$$

The main difference respect to the linear separable case is in the constraint on the non-negativity of the Lagrange multipliers of the dual problem,

$$0 \leq \alpha_i \leq C,$$

which set an upper limit of the value of the Lagrange multipliers. This aspect is also known as box constraint, as stated in [3].

The $C > 0$ is a penalty parameter (or regularization parameter) which controls the trade-off between the slack variable penalty (tolerance on training errors) and the margin and complexity of the model [3, 51, 38]. As stated in [15], the value of $C$ limit the influence of the outliers because without it the Lagrange multipliers would have a very large value. In [51] it is suggested to choose $C = \frac{1}{\nu N}$ where $0 < \nu \leq 1$ directly take into account the number of allowable misclassified point ($\nu$-SVM classifier). If the value of $C$ is set to infinity ($C = \infty$), the soft margin come back to the original version treated in the linear separable data case without any box constraint and, to distinguish it from the soft margin case, it is called hard margin [15].
It is worth noting that, after solved the dual problem, the optimal $\boldsymbol{w}_0$ and $b_0$ are known, and the classification of new input data $\overline{\boldsymbol{x}}$ can be performed only studying the sign function of the (4.2.2) which directly give the labels $y(\overline{\boldsymbol{x}})$:

$$y(\overline{\boldsymbol{x}}) = \text{sign}\left[\boldsymbol{w}_0 \cdot \overline{\boldsymbol{x}} + b_0\right] = \text{sign}\left[\sum_{i \in \text{SV}} \alpha_i y_i \boldsymbol{x}_i \cdot \overline{\boldsymbol{x}} + b_0\right]. \tag{4.2.24}$$

When the training data are separable but not in linear way in the input space, this is the case of non-linearly separable data. The idea is to map the input data to another space (feature space) usually with higher dimension [15] where the data are linearly separable. For greater clarity about the mapping concept, one can refer to the Figures 4.6 and 4.7. In these Figures are presented two examples where the data are separable in somehow but not in a linear way. Therefore, through a mapping function, the data are projected in a higher dimensional space in order to obtain a new dataset which is now linearly separable in that new space. After performing the SVM, the optimal hyperplane position is known but it lies in the new space. In order to find the non-linear separator in the original space, the points which belong to the optimal hyperplane have to be remapped with the inverse of the mapping function.

In the non-linear SVM, the transformation is performed by a Kernel Function which is a real-valued function of two arguments $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$ for $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$ typically symmetric ($K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_j, \boldsymbol{x}_i)$) and non negative ($K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$) [51]. The use of this type of function directly descends from the property which in SVM the classification depends only on the inner product of the training data, as a matter of fact, the Kernel function represents the inner product in the feature space [3]. This operation is also called Kernel Trick to the point of view of the algorithm [51]:

> *"Rather than defining our feature vector in terms of kernels, $\phi(x) = [K(x, x_1), ..., K(x, x_N)]$, we can instead work with the original feature vectors x, but modify the algorithm so that it replaces all inner products of the form $x \cdot x'$ with a call to the kernel function, $K(x, x')$. This is called the kernel trick. [...] Note that we require that the kernel be a Mercer kernel for this trick to work".*

Before the classification, one has to perform a non linear transformation $x \to \phi(x)$ from the input space to the feature space and then in the feature space one have to

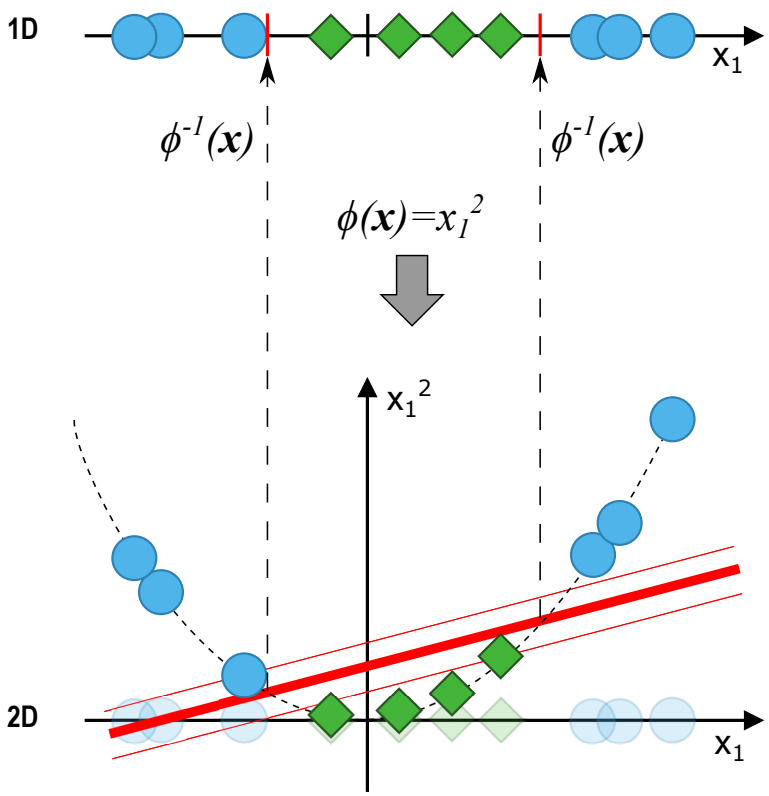



**Figure 4.6:** Example of mapping the data from a unidimensional space to a bi-dimensional space to transform the original non linearly separable dataset in a linearly separable dataset. The original data are mapped to a bi-dimensional space through an explicit mapping function $\phi(\boldsymbol{x}) = x_1^2$. In this new higher dimensional space, the data are now linearly separable and it is possible to apply the SVM to find the margins and the optimal hyperplane position (red solid lines). The intersections between the hyperplane and the mapping function (dashed line) are remapped through $\phi^{-1}(\boldsymbol{x})$ in order to find the position of the separator between the two classes in the original unidimensional space.

search the optimal hyperplane with maximal margin principle:

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x}_j) \cdot \boldsymbol{w} + b. \tag{4.2.25}$$

Generalizing equation (4.2.9), it is possible to write

$$\boldsymbol{w} = \sum_{i=1}^{N} y_i \alpha_i \phi(\boldsymbol{x}_i), \tag{4.2.26}$$

then it is possible to rewrite the (4.2.25) as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} y_i \alpha_i \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) + b. \tag{4.2.27}$$

Considering the general forms of the inner product in a Hilbert space [14]

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j), \tag{4.2.28}$$

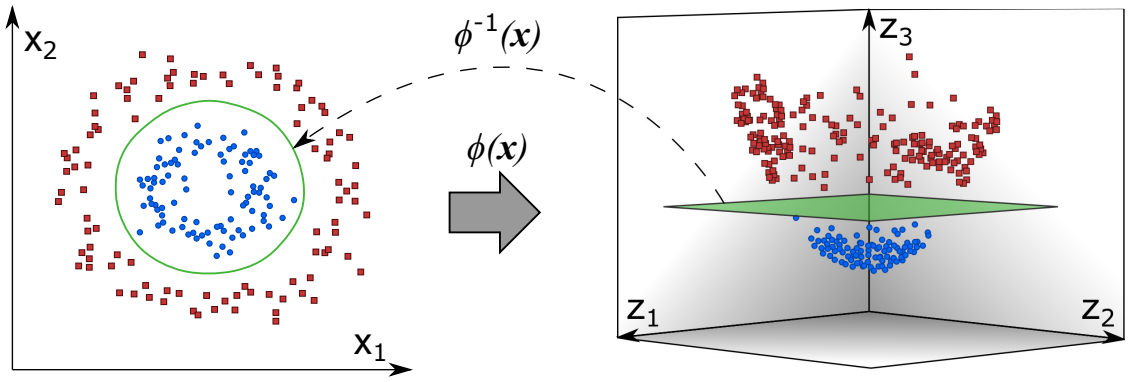



**Figure 4.7:** Example of mapping the data from a bi-dimensional space to a three-dimensional space in order to transform the original non linearly separable dataset in a linearly separable dataset. The original data are mapped to a three-dimensional space through an explicit mapping function. $\phi(\boldsymbol{x})$. In this new higher dimensional space, the data are now linearly separable and it is possible to apply the SVM to find the margins and the optimal hyperplane position (green plane on the right). The points which belong to the optimal hyperplane are remapped in the original bi-dimensional space through $\phi^{-1}(\boldsymbol{x})$, in order to find the non-linear separation boundary between the two classes in the original bi-dimensional space. Image taken by [65].

it is possible to write the decision function as the following sign function [38]

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\right). \tag{4.2.29}$$

In conclusion, since the SVM is based on inner product in feature space, one can avoid explicitly writing the transformation $\phi(\boldsymbol{x})$ of training data into the feature space, but one can operate directly defining a properly kernel function, as stated in [3]. This approach is called in [15] as implicit mapping or implicit embedding. In order to work with the kernel trick, in [51] is underlined the importance to choose a valid kernel i.e. it correctly represents the inner product in the feature space, hence it must be a Mercer kernel. According to [38], typical widely spread kernel functions are polynomial kernels

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + 1]^d \tag{4.2.30}$$

with $d \in \mathbb{N}$, Gaussian kernel [3, 15]

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2) \tag{4.2.31}$$

where $\sigma^2$ is called bandwidth. This latter kernel type is the isotropic form of radial basis kernel, RBF [51]. Multilayer perceptron kernels (MLP) or sigmoid kernels

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\gamma(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + r) \tag{4.2.32}$$

with $\gamma, r \in \mathbb{R}$ are also examples of kernel and their name is due to the fact that the same function is typically used in Neural Networks.

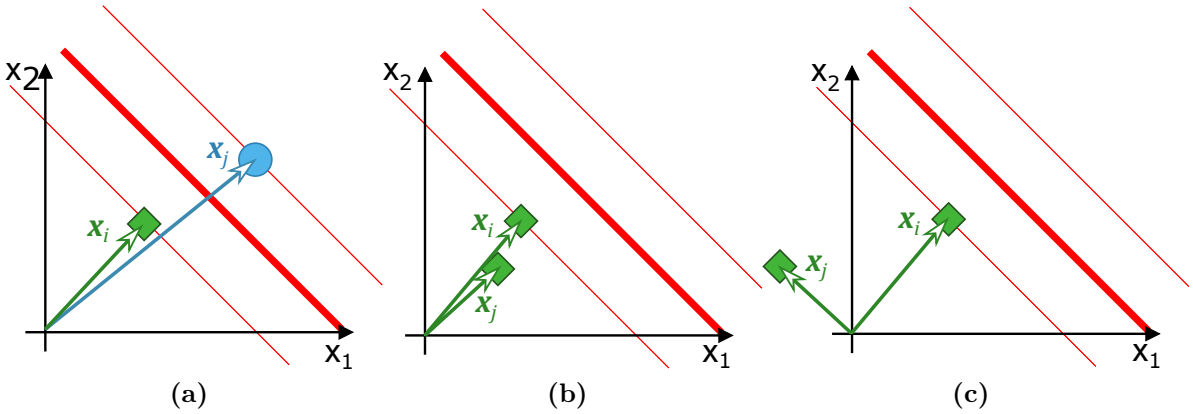



**Figure 4.8:** Inner product as a measure of similarity: the two vectors tend to maximize the margin in the left picture; in the center one, the two vectors are redundant for the SVM and do not add other information; on the right, the two orthogonal vectors do not count at all because their inner product is zero.

The SVM can also be adapted for multi-class classification problem e.g. adopting the approach "*one-versus-the-rest*". If the number of classes is $K > 2$, this method consists in training $K$ SVMs posing the $i$-th class under study as the positive class and all the other as the negative class, with $i = 1, 2, ..., K$. This method has some drawbacks therefore other techniques were developed to tackling this problem.

The SVM can also be adapted for regression problems minimizing a certain error function. For further readings about multi-class SVM and regression SVM, one can refer to [3, 51].

Before concluding the present section, it is worth noting that the inner product has a crucial importance in SVM. The inner product typically measure the similarity among the data as affirmed in [83]. In Figure 4.8, a graphical representation of the inner product as a measure of similarity in the bi-dimensional space is presented. If the two vectors belong to two different classes they tend to maximize the margin as demonstrated in (4.2.3), if the two vectors belong to the same class are redundant for the SVM and if the two vectors are orthogonal they do not count at all. Furthermore, also the kernel trick is based on a certain type of functions which try to generalize the concept of similarity of pairs of data from the input space to the feature space in which the kernel represents the inner product [3].

## 4.3 A new machine learning-based approach to handle constraints: PSO-SVM

In the proposed approach the aim is to integrate the classical PSO algorithm with SVM classification algorithm in order to separate feasible positions from the unfeasible ones. The main idea is to reduce the search space as also stated in [41].

Starting from the initial random population sampled with the LHS technique, it is possible to label feasible initial positions and the unfeasible ones with respect to the constraints evaluations. In general, through the SVM, a hyperplane which linearly divides data is constructed in any higher-dimensional space (feature space) and it is possible to reconstruct a non-linear boundary in the starting design space. Indeed, this is a way to reduce the search space because, after the learning phase, the trained SVM model can predict if new unseen points are in the feasible design region or in the unfeasible one. In order to get an accurate boundary of the feasible region even from the beginning, it would be necessary to sample a very huge swarm size. However, considering the PSO mechanisms, at every generation, the swarm moves in a new different configuration, leading to new available positions. Therefore, labelling these new positions, it is possible to increase the knowledge database and train a new improved SVM model. Therefore, during the evolutionary stage of PSO, a new boundary is calculated at each generation considering all the visited labelled positions until the current iteration. This incremental approach has been adopted in the new proposed PSO-SVM and it has been implemented in Matlab® code. As above mentioned, in the various PSO framework, constraint handling techniques do not work very well with equality constraints but they are more suitable for inequality constraint only. Therefore, the problem has to be expressed in the form of (2.1.2) without equality constraints. For a maximization problem with objective function $f(\boldsymbol{x})$, it is sufficient consider $-f(\boldsymbol{x})$ to transform it into a minimization problem.

In Figure 4.9 the flowchart of the proposed algorithm is shown. At first, the initialization phase of the swarm adopts the LHS technique to generate the initial population with a minimum correlation between samples (see also Monti et al. [49]). After that, this initial swam is labelled into the design space considering all the inequality constraints. Since these latter are defined as $g_q \leq 0$, if at least one of them is greater than zero the entire $i$-th n-tuple (which is a single individual, a candidate solution) is labelled as unfeasible ($y_i = -1$) otherwise it is labeled as feasible ($y_i = +1$). Unfortunately, sometimes, the SVM really struggles to work properly, e.g. with a very narrow search space. In fact, the preservation of the feasible points only, into a wide unfeasible space, leads to instabilities such as overfitting or underfitting problems of the SVM. To improve the performance of the algorithm, a relax constraint function $\psi_i(k)$ is defined, wherein the subscript $i$ refers to $i$-th constraint whereas the $k$ refers to the current generation. This approach leads to enlarging the real feasible space to a fictitious wider one using a relaxation of the constraints. This means that the real constraints are “moved" from their actual position to a fictitious one through a proper choice of $\psi_i(k)$ which is not trivial to generalize. This procedure acts as substituting the original inequality constraints with the following relaxed constraints:

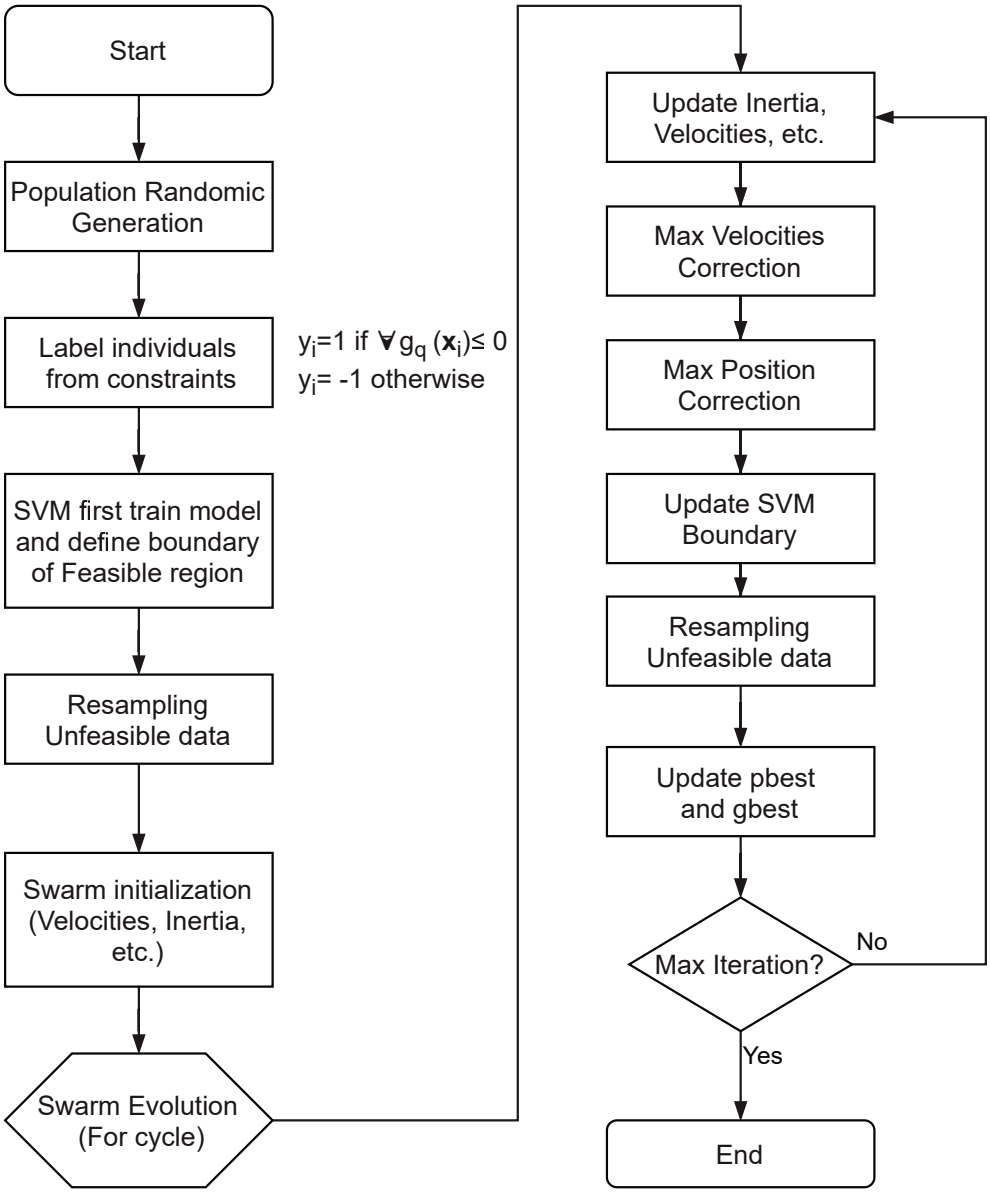



**Figure 4.9:** General Flowchart of the proposed PSO-SVM algorithm.

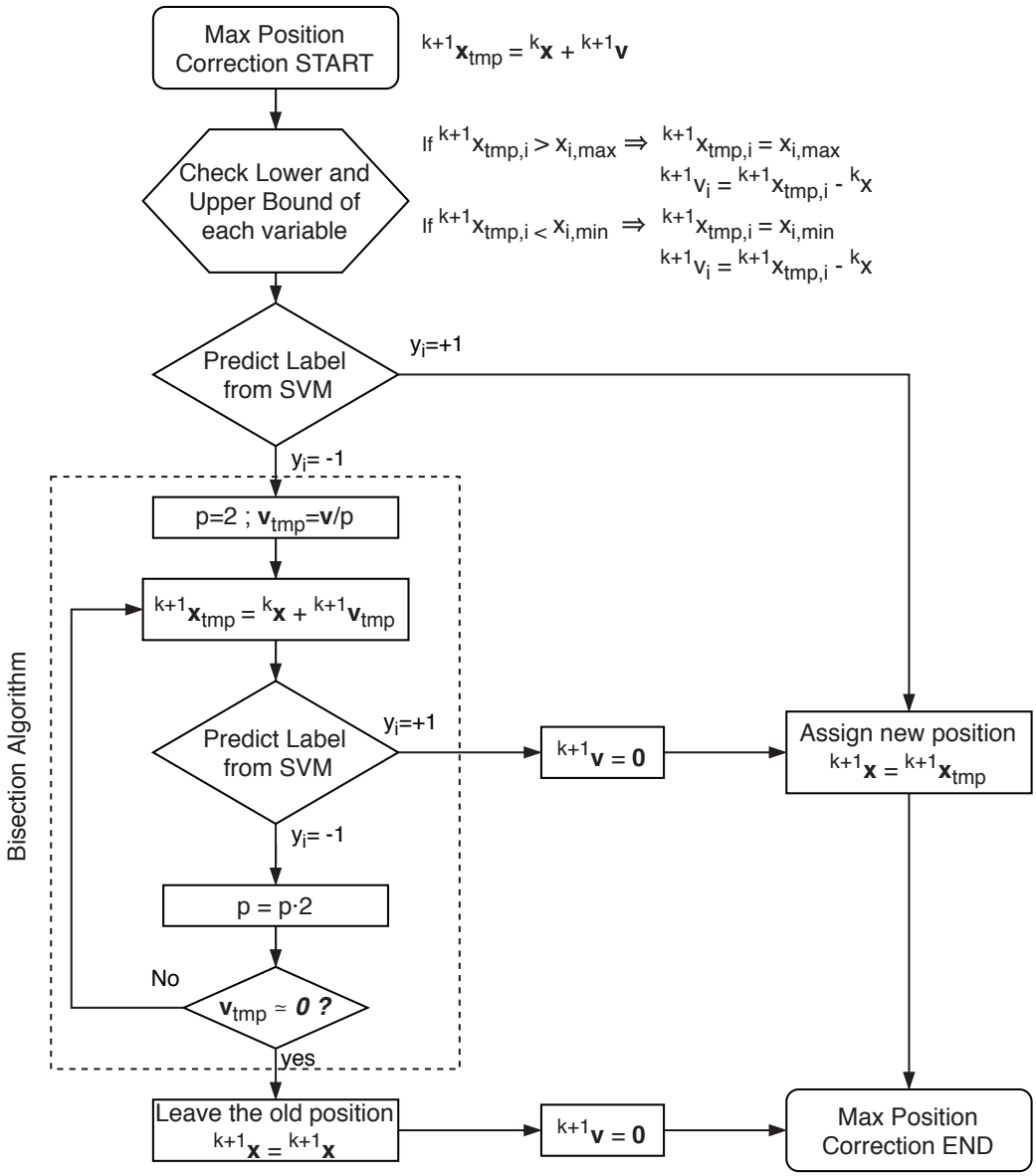



**Figure 4.10:** Flowchart of the "Max Position Correction" block. The bisection algorithm part is underlined by the dashed line.

$$g_{i,k}(\boldsymbol{x}) \leq \psi_i(k) \qquad \Longrightarrow \qquad g_{i,k}(\boldsymbol{x}) - \psi_i(k) \leq 0. \tag{4.3.1}$$

The $\psi_i(k)$ factor is tuned on the standard deviation $\sigma_{u,k}$ of the amount of all unfeasible points detected in all generations until the current one. This approach allows a better exploration in the first generations and then it could be forced to zero implementing a decreasing function which directly depends on the current generation $k$ and the number of maximum generations $k_{max}$. Defining a further reduction factor $\lambda$ of the standard deviation directly chosen by the user, the following relaxation functions are implemented:

- Constant Relax:

$$\psi_i(k) = \lambda\sigma_{u,k};$$

- Piecewise Constant Relax:

$$\psi_i(k) = \begin{cases} \lambda\sigma_{u,k}, & \text{if } k \leq k_{max}/2 \\ 0, & \text{otherwise;} \end{cases}$$

- Linear decreasing Relax:

$$\psi_i(k) = \lambda\sigma_{u,k} - \frac{\lambda\sigma_{u,k}}{k_{max}}k;$$

- Piecewise Linear decreasing Relax:

$$\psi_i(k) = \begin{cases} \lambda\sigma_{u,k} - \frac{2\lambda\sigma_{u,k}}{k_{max}}k, & \text{if } k \leq k_{max}/2 \\ 0, & \text{otherwise;} \end{cases}$$

- Parabolic decreasing Relax:

$$\psi_i(k) = \lambda\sigma_{u,k} - \frac{\lambda\sigma_{u,k}}{k_{max}^2}k^2;$$

- Piecewise Parabolic decreasing Relax:

$$\psi_i(k) = \begin{cases} \lambda\sigma_{u,k} - \frac{4\lambda\sigma_{u,k}}{k_{max}^2}k^2, & \text{if } k \leq k_{max}/2 \\ 0, & \text{otherwise;} \end{cases}$$

The reduction factor $\lambda$ have to be chosen by the user in order to get the best trade-off between performances, exploration vs exploitation, computational speed. In Matlab® the training phase of the SVM model is performed by the function `fitcsvm`. The user has to define two important parameters which can affect the SVM model. The first is the type of kernel function to use (we especially adopt the `rbf` Gaussian kernel function). The second parameter is the box-constraint which takes value from 1 to `inf` which affects the softness of the maximal margin. In fact, `fitcsvm` naturally implements the soft margin with the $C = 1$. If we set $C =$`inf` then we are using a hard maximal margin but this can lead to numerical instabilities if the data are

not perfectly separable. In any case, using a proper hard margin create a higher computational effort and it very slows down the algorithm. The proposed algorithm implements SVM with soft margin with regularization parameter $C = 100$. Indeed, respect to a hard margin, to speed up the algorithm it is numerically convenient to adopt a high box-constraint value, e.g. between 100 and 1000, allowing a certain misclassification rate but assuring all the possible solutions lie in the feasible space boundary respecting the constraints with a certain engineering tolerance.

After the training phase, the unfeasible points of the initial population have to be re-sampled in the feasible region. The Matlab function `predict` allows to get the label of new entry data in the design space based on the trained SVM model. So, the unfeasible points are randomly sampled with LHS technique in the entire design space and they are discarded and sampled again until their label becomes $y_i = +1$.

At this point, the initial velocity of the particles is randomly sampled always using LHS and then the evolutionary phase of the algorithm can begin. The velocity for the next generation is calculated as in equation (3.1.1) but it is necessary to check if it respect the allowable maximum velocity as in (3.1.5) ("Max Velocities Correction" block in Figure 4.9). The next block "Max Position Correction" is a fundamental step because it is the assessment of the feasibility of the position given by (3.1.2) and, for greater clarity, this block is expanded in Figure 4.10. The temporary positions are computed with equation (3.1.2). Then, considering the side constraints, a first possible correction of the temporary positions can be performed in order to maintain the particles inside or at least on the edges of the hyper-rectangle design space $\Omega$. If the temporary position label is $y_i = +1$, the new temporary position lies in the feasible region and, therefore, it is accepted and stored. Otherwise, if the label is negative, a simple bisection approach is performed. The velocity vector which leads to the temporary position is firstly halved by a factor $p = 2$ obtaining a new temporary position. If the label of this new temporary position becomes positive, this new position is accepted for the next generation. Otherwise, if it is still negative, the algorithm will increment $p$ doubling it, getting a new temporary velocity vector which leads to a new temporary position to label and so on. When all the elements of the temporary velocity vector tend to zero, it is possible to leave the original position for the next generation without moving that particle. In this way, the particle can move only if the destination position is into the feasible region, otherwise it does not move anywhere during the current generation.

At the end of each evolutionary stage, new positions are available to increase the knowledge database to train a new better SVM model and update the feasible region boundary. As before, it is necessary to label the new points from the constraint expressions and start the new training phase.After that, a new boundary is defined but some of the points which were inside the previous boundary now may lie outside of the new region. In this case, it is necessary to re-sample these points as before using LHS until their labels become positive. Despite this latter naive approach may slow the algorithm, it is beneficial for the diversity of the population, allowing new starting search positions. The update of the gbest is performed only on the real feasible points respect to the original constraints and not the relaxed ones. This procedure ensures the algorithm performs good results also using a constant relax constraint function during the generations. In fact, at each generation, it will

update the global optimum point (gbest) looking to the minimum objective function value only of those points which lie into the real feasible region. A further strategy to improve the behaviour of the PSO is reducing the maximum possible velocity range of the particles, governed by $\gamma$, updating this latter during the generations. In the proposed algorithm it is set $\gamma/2$ starting from $k_{max}/3$ then it is set to a minimum value e.g. $\gamma = 0.1$ in the last generations from $2k_{max}/3$ to $k_{max}$, where $k_{max}$ represents the max allowable generations. The stopping criterion of the algorithm is the achievement of a maximum number of iterations. It is possible which there exist an entire front of optimal solutions so one has always to check the convergent history of best solutions during the post-processing phase.

In the next chapter, the proposed approach is firstly tested on some numerical mathematical literature problems which statements are reported in the Appendix. After that, two structural optimization problems are developed. To make some comparison of the results, in these last examples the objective function value is compared with the genetic algorithm of Matlab® and the PSO with penalty approach provided from the code proposed by [42].

# Chapter 5

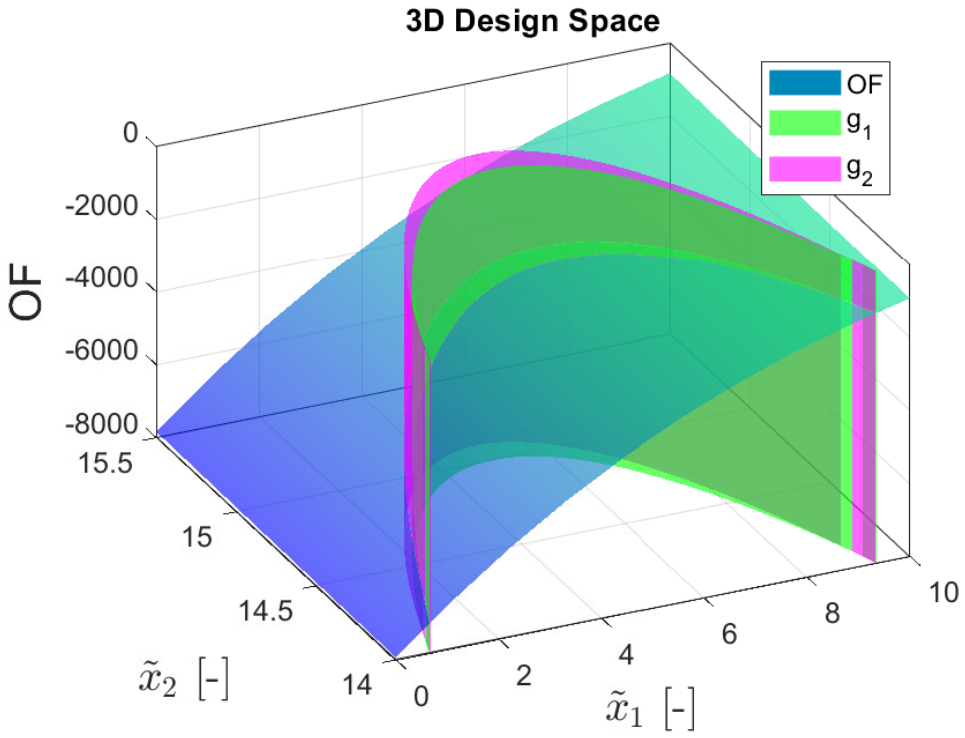
# Case Studies

In order to verify the convergence of the new code in terms of objective function, in the present chapter, the proposed PSO-SVM algorithm is adopted in two numerical literature benchmark examples and it is compared with a PSO with penalty function code and with a genetic algorithm (GA) code. In the end part of the present chapter, the proposed PSO-SVM is adopted into the structural optimization field analyzing two different examples. The first one is concerning the size optimization of a simply supported beam with constant rectangular cross section subjected to a constant load condition. Despite its theoretical simplicity, this problem was selected because, reducing the design variables to only two, it is possible to give a graphical representation of the problem in the design space. As a matter of fact, the graphical obtained results highlight the generality of the SVM to handle piecewise non-linear constraints with a good approximation in the neighbourhood of the optimal point and a roughly approximation elsewhere. The second example is regarding shape and size optimization of a warren truss plane beam which can be used in the design of a steel truss arc bridge. The PSO-SVM optimal results are compared in terms of total weight with the result obtained in [23] where differential evolutionary algorithm (DEA) is adopted. The exact solution needs to be industrialized choosing a real existing profile dimensions both with a simple rounding-off of the optimal solution and with a more refined post-processing approach. Making a comparison between these latter industrial solutions the total weight of the structure does not change significantly and a simple rounding-off approach can be easily adopted by the designer without jeopardizing the entire optimization process.

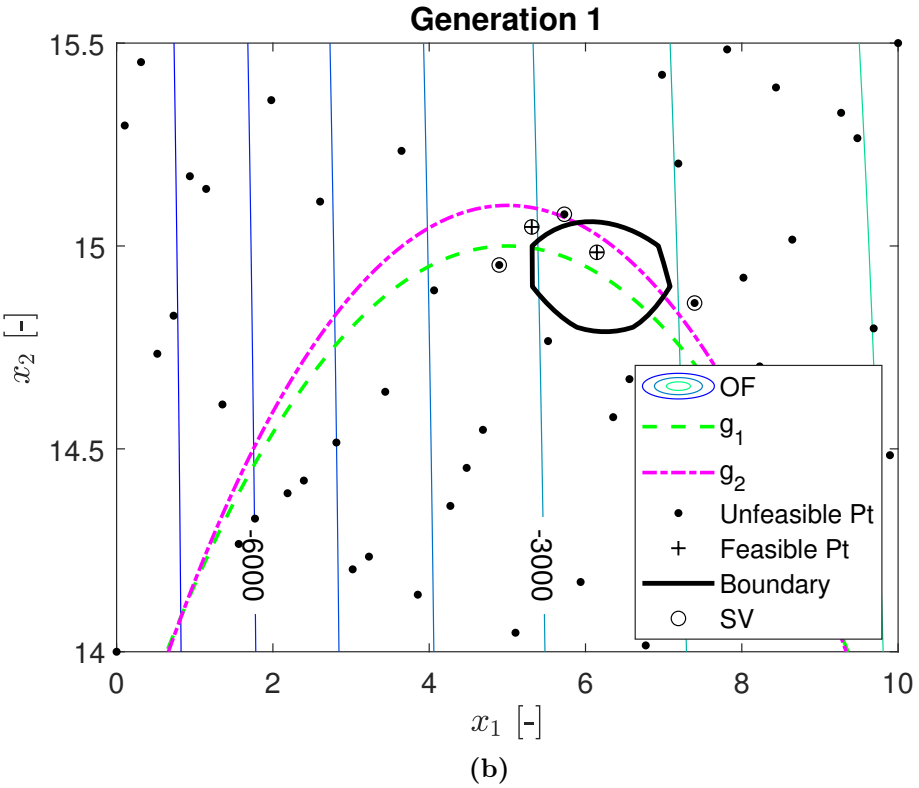
## 5.1 Numerical Benchmarks Problems

### 5.1.1 Numerical Example 1: Sickle Problem [54]

The first numerical example (from literature [54]) is the benchmark test 1 stated in the Appendix also known as Sickle function. Since this optimization problem has two design variables, it can be depicted in a graphical representation in the bi-dimensional design space of the search process performed by PSO-SVM algorithm. In Figure 5.1 (a), the objective function and the constraints are graphically represented as a 3D graph and it is possible to notice that the objective function

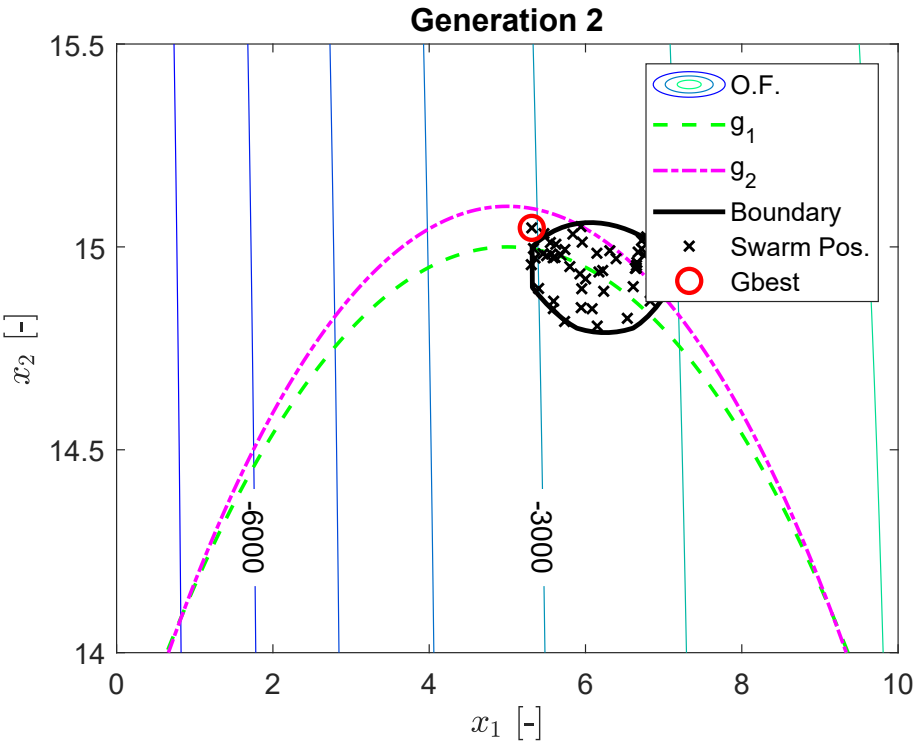



**(a)**

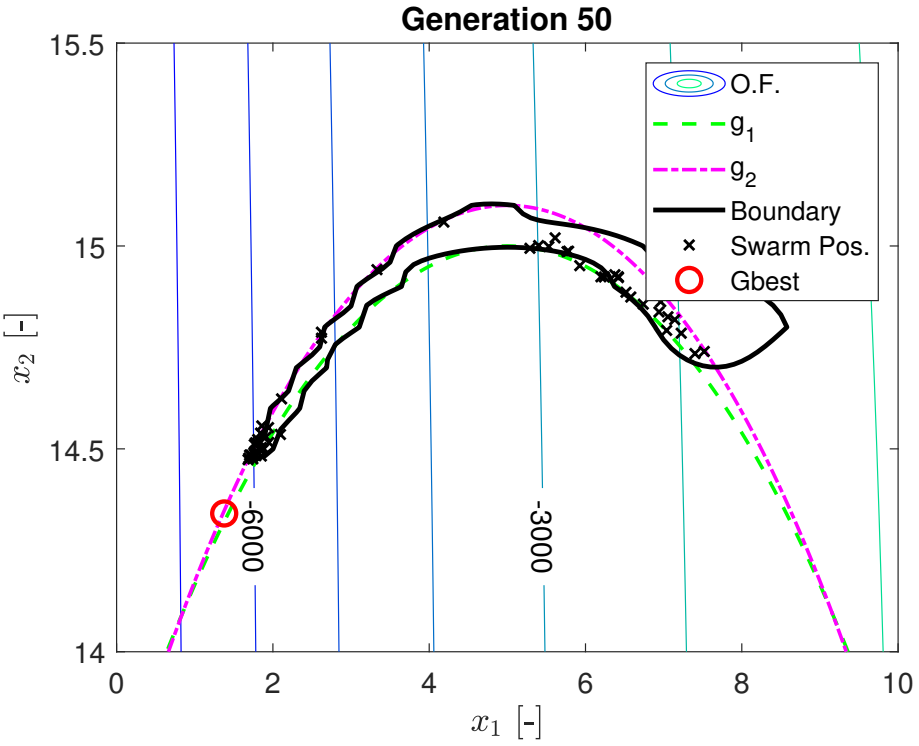



**(b)**

**Figure 5.1:** Example 1 (Sickle Problem [54]), case No relax constraints function; (a) Three-dimensional graph of Sickle problem design space; (b) Generation 1.

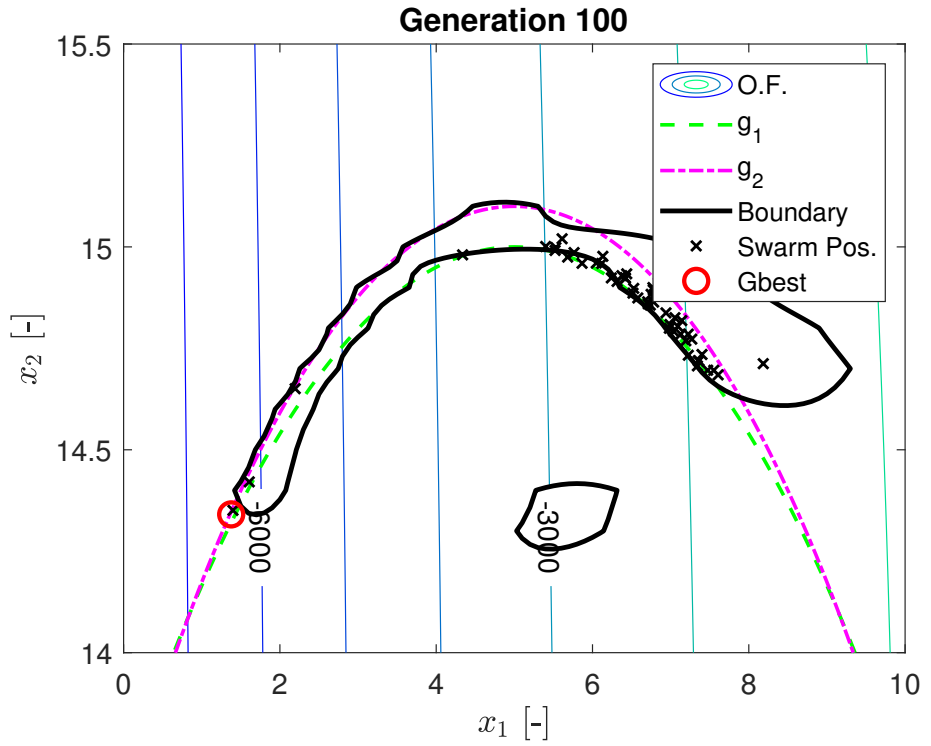



(a)

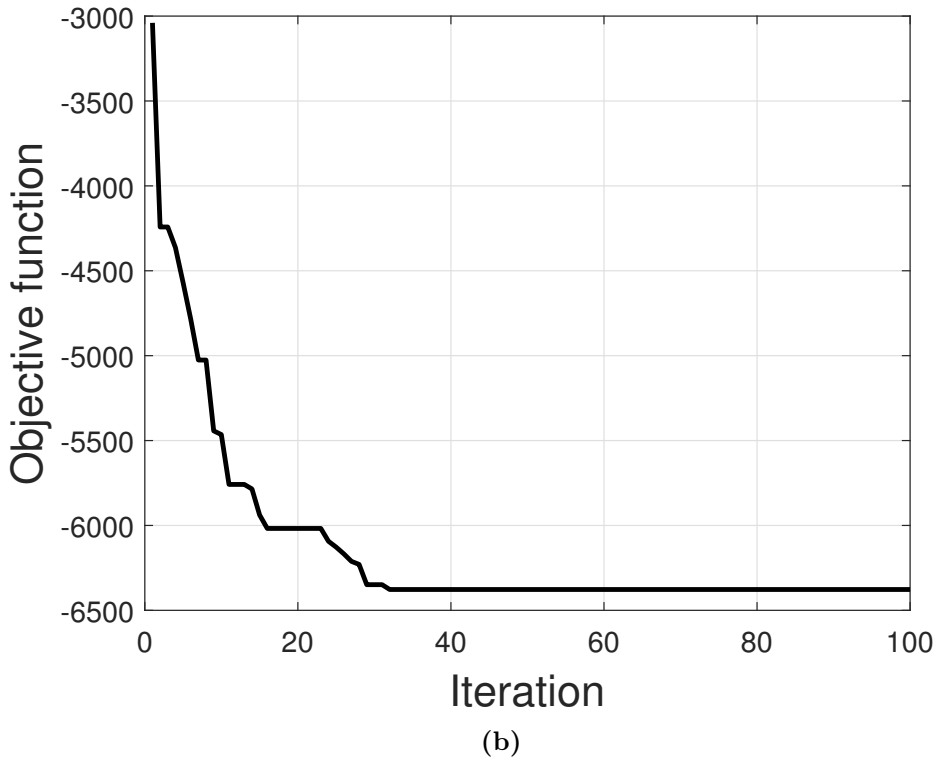



(b)

**Figure 5.2:** Example 1 (Sickle Problem [54]), case No relax constraints function; (a) Generation 2; (b) Generation 50.

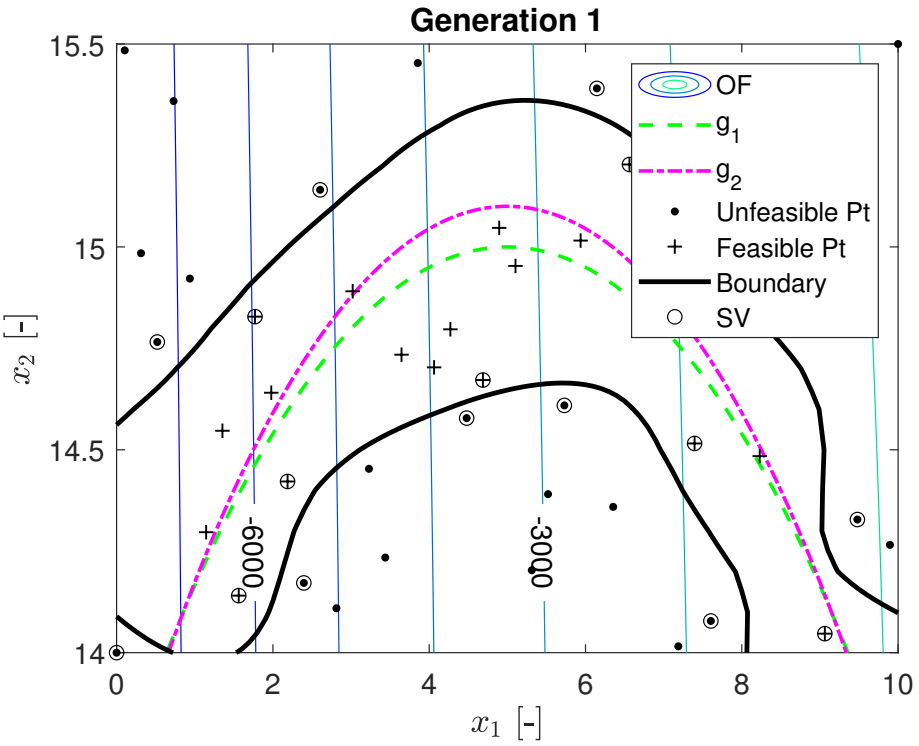



(a)

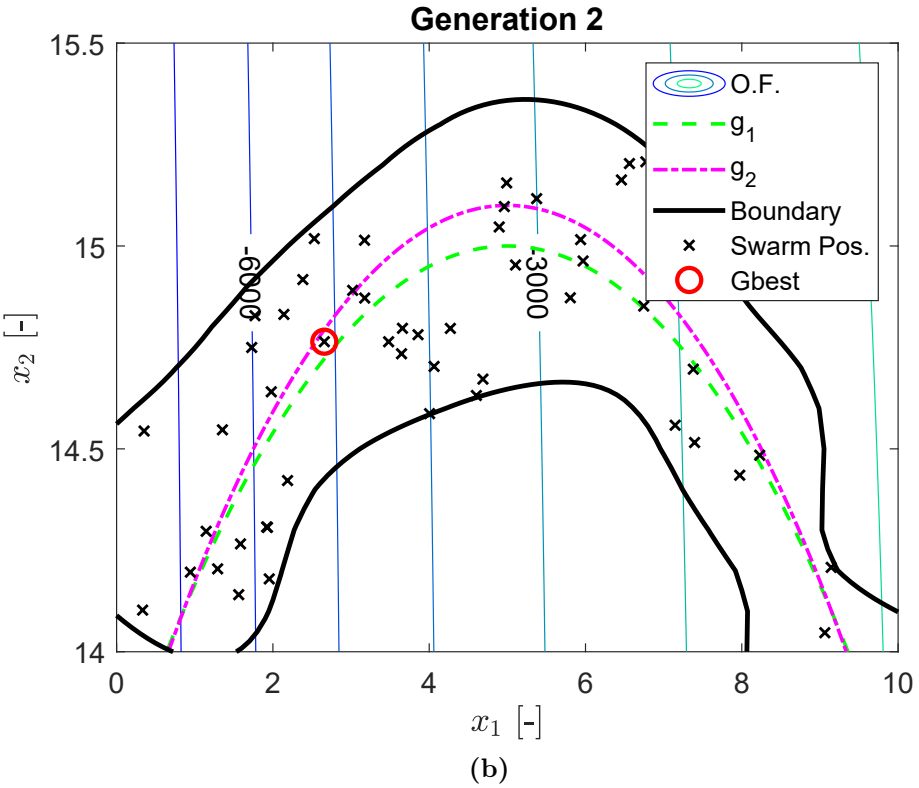



(b)

**Figure 5.3:** Example 1 (Sickle Problem [54]), case No relax constraints function; (a) Generation 100; (b) Objective function history.

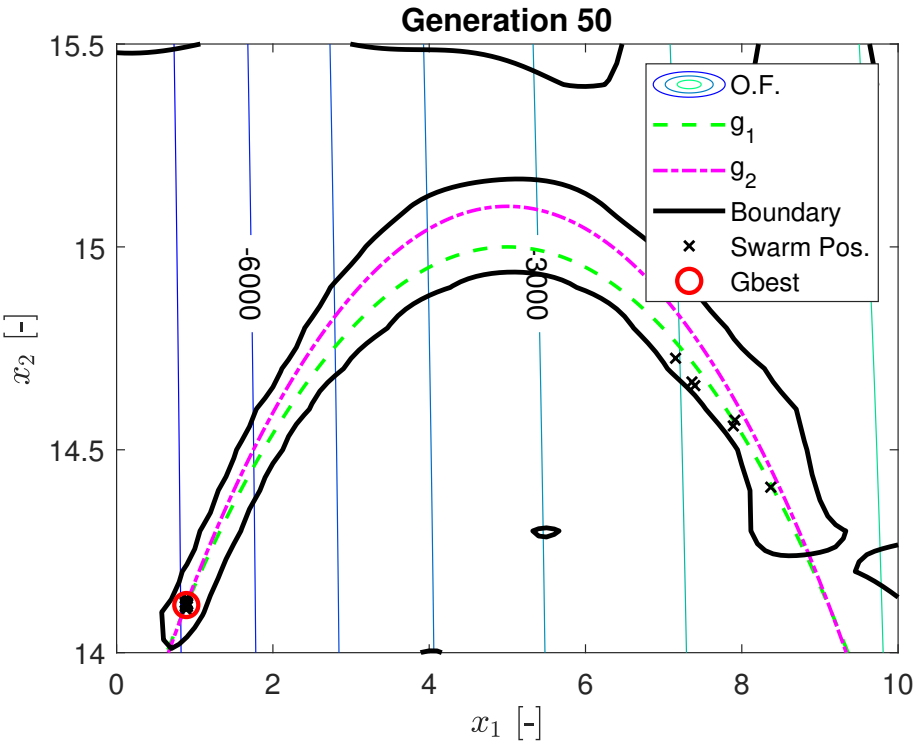



(a)

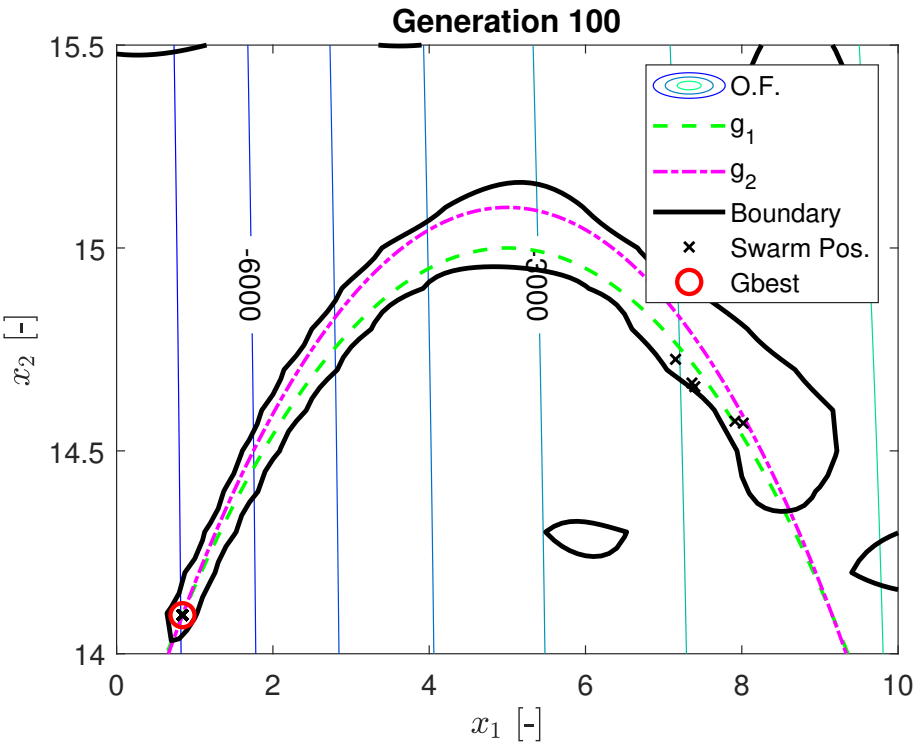



(b)

**Figure 5.4:** Example 1 (Sickle Problem [54]), case constant relax constraints function; (a) Generation 1; (b) Generation 2.




**(a)**




**(b)**

**Figure 5.5:** Example 1 (Sickle Problem [54]), case constant relax constraints function; (a) Generation 50; (b) Generation 100.

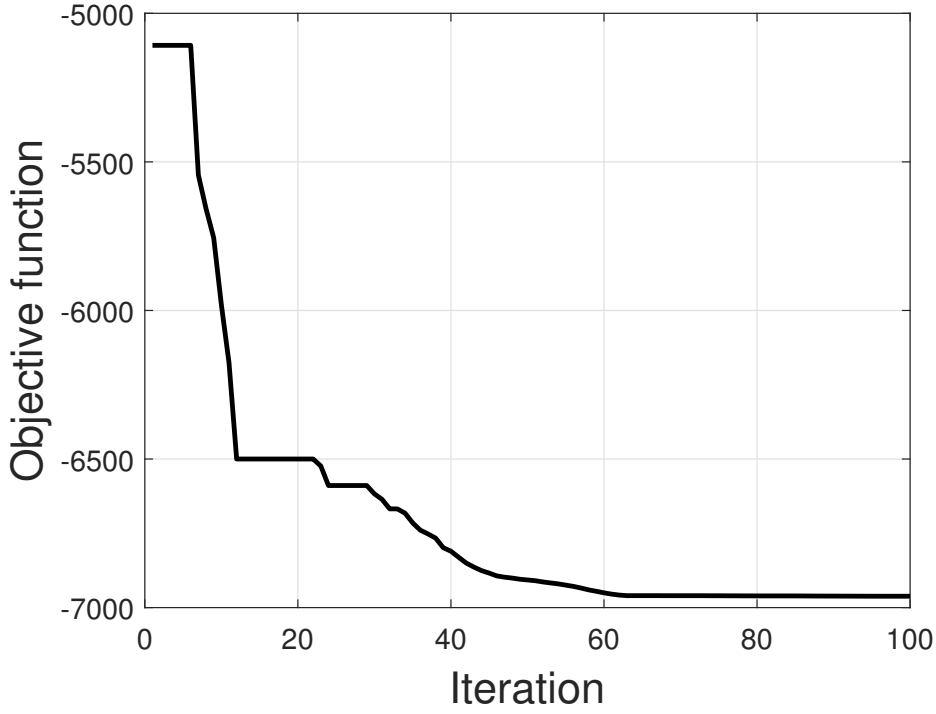



**Figure 5.6:** Example 1 (Sickle Problem [54]), case constant relax constraints function, objective function history.

is substantially planar in the search space. The projection on the design variables plane of the feasible region is really narrow because it is formed by the space between the two constraints parabolas. For this problem, a population size of 50 individuals is set and in Figure 5.1 (b) the first random generation is shown and the support vectors are emphasized. After that, the unfeasible points are re-sampled until all the population falls inside the SVM-based boundary. At each generation, new points are added to the SVM training data and the boundary is improved. Since the search space is really narrow, running the algorithm with no relax constraint function it results in poor performance in defining the SVM boundary (black solid line) with respect to the actual one (dashed lines) as shown in Figures 5.2 and 5.3. It is possible to improve the performance in terms of objective function decreasing history using the relax of constraints, as e.g. constant relax function shown in Figures 5.4-5.5-5.6. In this latter example, the reduction factor of standard deviation of the unfeasible point is chosen as $\lambda = 0.5$. The PSO-SVM performances using constant relax function is compared with other algorithms in Table 5.1. The comparison is done in terms of objective function value to the optimal solution after 50 runs with $k_{max} = 100$ each run and collecting mean value $\mu$, standard deviations $\sigma$, best OF and worst OF. As one can check, the convergence of the new proposed method is satisfied as the objective function value is getting close to the global optimum with a little standard deviation as the other existing methods.

**Table 5.1:** Numerical Example 1 (Sickle Problem [54]), comparison PSO-SVM, GA and PSO-penalty.

| | PSO-SVM | PSO-Penalty | GA |
|---|---|---|---|
| $\mu_{x_1}$ | 0.8433 | 0.8721 | 0.8502 |
| $\sigma_{x_1}$ | 0.0004 | 1.121e-15 | 0.0172 |
| $\mu_{x_2}$ | 14.0952 | 14.1091 | 14.0986 |
| $\sigma_{x_2}$ | 0.0002 | 8.972e-15 | 0.0087 |
| $\mu_{OF}$ | -6.9614e+03 | -6.9291e+03 | -6.9537e+03 |
| $\sigma_{OF}$ | 0.4609 | 4.59e-12 | 19.2968 |
| Best OF | -6.9595e+03 | -6.93e+03 | -6.9618e+03 |
| Worst OF | -6.96e+03 | -6.93e+03 | -6.8547e+03 |

### 5.1.2 Numerical Example 2: five design variables optimization problem

The second numerical example statement reported in the Appendix is a literature benchmark test optimization problem with five design variables and six constraints. In this example, a comparison between different histories of the objective function using different relax constraint functions is performed. As shown in Figure 5.7, all the examples give a good result and tend to converge to the exact solution with a different decreasing rate. In general, one can notice that in piecewise functions, the algorithm generally boosts the exploration instead of the exploitation that is usually performed in the second half with a zero relax coefficient. This feature is important because can affect the performance of the proposed algorithm with different kind of problems and the user need to try different relax functions in order to find the most suitable for this kind of problem. In Tables 5.2 and 5.3, a comparison with the GA and the PSO-Penalty is performed in terms of objective function value and optimal design points running the codes 50 times with $k_{max} = 100$ each run

**Table 5.2:** Numerical Example 2, comparison among PSO-SVM with different relax constraint fuctions (check the Appendix and [67]).

| Comparison among Constraint Relax Functions in PSO-SVM | | | | | | | |
|---|---|---|---|---|---|---|---|
| Design Var. | No relax | Const. relax | Piecewise const. | Lin. relax | Piecewise lin. | Parabolic relax | Piecewise Par. |
| $\mu_{x_1}$ | 78.0004 | 78.0086 | 78.0026 | 78.0009 | 78.0006 | 78.0027 | 78.0077 |
| $\sigma_{x_1}$ | 0.0013 | 0.0243 | 0.0120 | 0.0038 | 0.0041 | 0.0100 | 0.0526 |
| $\mu_{x_2}$ | 33.0062 | 33.0040 | 33.0056 | 33.0104 | 33.0107 | 33.0740 | 33.0052 |
| $\sigma_{x_2}$ | 0.0158 | 0.0183 | 0.0143 | 0.0438 | 0.0322 | 0.2274 | 0.0158 |
| $\mu_{x_3}$ | 30.0027 | 29.9985 | 30.0037 | 30.0009 | 30.0038 | 30.0366 | 30.0027 |
| $\sigma_{x_3}$ | 0.0101 | 0.0105 | 0.0125 | 0.0235 | 0.0178 | 0.1217 | 0.0160 |
| $\mu_{x_4}$ | 44.6569 | 44.2360 | 44.3643 | 44.9858 | 44.8002 | 44.0814 | 44.6662 |
| $\sigma_{x_4}$ | 1.5986 | 2.9630 | 2.0027 | 0.0617 | 1.0734 | 3.2862 | 1.3233 |
| $\mu_{x_5}$ | 36.8981 | 37.0755 | 37.0136 | 36.7671 | 36.8360 | 37.0446 | 36.8931 |
| $\sigma_{x_5}$ | 0.6444 | 1.1956 | 0.8036 | 0.0652 | 0.4310 | 1.3843 | 0.5324 |
| $\mu_{OF}$ | -3.0655e+04 | -3.0644e+04 | -3.0647e+04 | -3.0664e+04 | -3.0659e+04 | -3.0634e+04 | -3.0655e+04 |
| $\sigma_{OF}$ | 43.0187 | 78.9205 | 53.7773 | 3.9918 | 28.9931 | 87.9967 | 36.2059 |
| Best OF | -3.0666e+04 | -3.0666e+04 | -3.0666e+04 | -3.0666e+04 | -3.0666e+04 | -3.0666e+04 | -3.0666e+04 |
| Worst OF | -3.0375e+04 | -3.0185e+04 | -3.0449e+04 | -3.0643e+04 | -3.0468e+04 | -3.0186e+04 | -3.0452e+04 |

**Table 5.3:** Numerical Example 2, results from PSO-SVM without relax constraints, PSO-Penalty and GA (check the Appendix and [67]).

| | PSO-SVM | PSO-Penalty | GA |
|---|---|---|---|
| $\mu_{x_1}$ | 78.0004 | 78 | 78.0004 |
| $\sigma_{x_1}$ | 0.0013 | 0 | 0.0024 |
| $\mu_{x_2}$ | 33.0062 | 33 | 34.2398 |
| $\sigma_{x_2}$ | 0.0158 | 0 | 0.7052 |
| $\mu_{x_3}$ | 30.0027 | 29.9967 | 30.801 |
| $\sigma_{x_3}$ | 0.0101 | 2.15e-14 | 0.378 |
| $\mu_{x_4}$ | 44.6569 | 45 | 45 |
| $\sigma_{x_4}$ | 1.5986 | 0 | 0 |
| $\mu_{x_5}$ | 36.8981 | 36.7736 | 34.8023 |
| $\sigma_{x_5}$ | 0.6444 | 0.00e+00 | 0.9057 |
| $\mu_{OF}$ | -3.0655e+04 | -3.0665e+04 | -3.0531e+04 |
| $\sigma_{OF}$ | 43.0187 | 1.47E-11 | 65.8031 |
| Best OF | -3.0666e+04 | -3.0665e+04 | -3.0660e+04 |
| Worst OF | -3.0375e+04 | -3.0665e+04 | -3.0378e+04 |

and collecting mean value $\mu$, standard deviation $\sigma$, worst and best. Also with this more complex optimization problem, the convergence of the new proposed method is satisfied getting an objective function value close to the global optimum with a little standard deviation as the other existing methods.

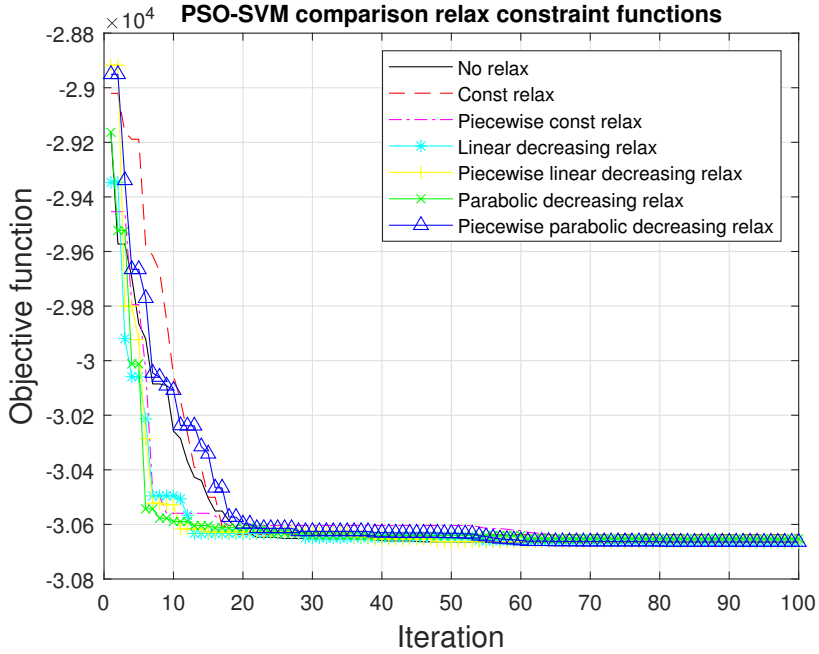



**Figure 5.7:** Numerical Example 2: Objective value history comparison among different relax constraint functions for a single run.

## 5.2 Structural optimization Problems

### 5.2.1 Structural Example 1: simply supported beam

In Figure 5.8 it is considered an ideal simply supported beam of length $L$ with a constant cross section $A = b \cdot h$ loaded with a distributed constant load $q$ which is supposed to be much grater than the self weight for the sake of simplicity. The aim is to minimize the weight of this structure respecting the tensional constraints and maximum deflection constraint due to only the $q$ load. The self weight is proportional to the volume $V$ of the material as stated by [76]. The objective function is $f(\boldsymbol{d}) = \rho V = \rho AL$, where $\boldsymbol{d}$ is the design vector and $\rho$ is the material density which is supposed to be constant. In this case, only stress constraints on normal stress $\sigma$, tangential stress $\tau$ and on the maximum deflection $v(z)$ are considered. This is a typical sizing optimization problem. The design vector $\boldsymbol{d}^T = \{d_1, d_2\}$ contains the design variables which are changed during the optimization process, i.e. in this case $d_1 = b$, $d_2 = h$. Since the cross section must be greater than zero as well as the cross sectional dimensions, it implies the presence of a new constraint to satisfy. Performing an elastic analysis, the maximum allowable stress is the yielding stress $\sigma_y$ and it is possible to use the Navier Formula and the Jourawsky Formula for the normal and tangential stress respectively. The maximum moment

$$M_{Ed}(z = \frac{L}{2}) = \frac{qL^2}{8}$$

is in the middle span and the maximum shear force

$$V_{Ed} = \frac{qL}{2}$$

is in correspondence of the supports $z = 0$, $z = L$. Recalling the elastic resistance modulus for a rectangular section

$$W_{el} = \frac{bh^2}{6} = \frac{d_1 d_2^2}{6}$$

and using the Navier formula it is possible to write the maximum normal stress in the middle span as

$$\sigma\Big(z = \frac{L}{2}\Big) = \frac{M_{Ed}}{W_{el}} = \frac{3}{4}\frac{qL^2}{d_1 d_2^2}. \tag{5.2.1}$$

Using the Jourawsky formula, it is possible to write the maximum tangential stress in the middle of cross section $y = 0$ (parabolic tangential stress diagram on a rectangular section) as

$$\tau(z = 0, z = L) = \frac{V_{Ed} S_x^*(y = 0)}{I_x b} = \frac{3}{2}\frac{V_{Ed}}{bh} = \frac{3}{4}\frac{qL}{d_1 d_2}. \tag{5.2.2}$$

In order to take into account both normal and tangential stresses it is necessary to refer to a yield criterion. In this case, the Von Mises yield criterion is adopted:

$$\sqrt{\sigma^2(z) + 3\tau^2(z)} \leq \sigma_{id} \tag{5.2.3}$$

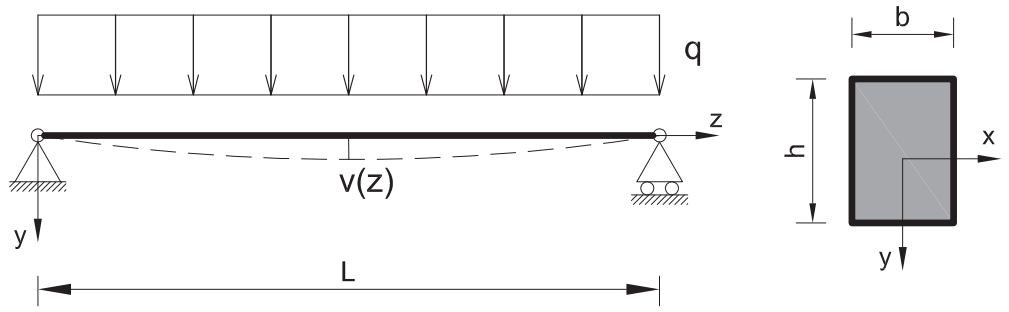



**Figure 5.8:** Problem formulation: simply supported beam with constant cross section.

Substituting the (5.2.1) and the (5.2.2) into (5.2.3) respectively it is possible to obtain the two expression of the stress constraints in middle span $z = L/2$ (pure moment) and in $z = 0, L$ (pure shear):

$$\frac{3}{4}\frac{qL^2}{d_1 d_2^2} \leq \sigma_{id}, \tag{5.2.4}$$

$$\frac{3}{4}\frac{qL}{d_1 d_2} \leq \frac{\sigma_{id}}{\sqrt{3}}. \tag{5.2.5}$$

The maximum deflection $v(z = \frac{L}{2})$ can be calculated using the virtual work principle obtaining

$$v\Big(z = \frac{L}{2}\Big) = \frac{5}{384}\frac{qL^4}{EI_x} = \frac{5}{32}\frac{qL^4}{Ed_1 d_2^3}. \tag{5.2.6}$$

The complete statement of the optimization problem is the following:

$$\begin{aligned} \min \quad & f(d_1, d_2) = d_1 d_2, \\ \text{s.t.} \quad & d_1 > 0,\ d_2 > 0, \\ & \frac{3}{4}\frac{qL^2}{d_1 d_2^2} - \sigma_{id} \leq 0, \\ & \frac{3}{4}\frac{qL}{d_1 d_2} - \frac{\sigma_{id}}{\sqrt{3}} \leq 0, \\ & \frac{5}{32}\frac{qL^4}{Ed_1 d_2^3} - v_{max} \leq 0, \end{aligned} \tag{5.2.7}$$

where the constant $\rho L$ is dropped by the objective function as stated in similar problems analyzed in [9], $\sigma_{id}$ is the ideal Von Mises normal stress and $v_{max}$ is the maximum deflection admissible by reference design codes i.e. in this case it is fixed to $v_{max} = L/250$. It is possible to define the fixed variables vector $\boldsymbol{b}^T = \{b_1, b_2, b_3, b_4, b_5\}$ which contains problem data which not change during the optimization process, i.e. in this case $b_1 = q$, $b_2 = L$, $b_3 = \sigma_{id}$, $b_4 = E$, $b_5 = v_{max}$. Since the amount of fixed parameters, it is more convenient work with a dimensionless form of the same problem. Posing $\tilde{d}_1 = b/L$ and $\tilde{d}_2 = h/L$, the new dimensionless objective function become

$$\tilde{f}(\tilde{d}_1, \tilde{d}_2) = f(d_1, d_2)/L^2 = \tilde{d}_1 \tilde{d}_2,$$

whilst the normal stress constraint (5.2.4), the shear constraint (5.2.5) and the deflection constraint (5.2.6) become respectively as

$$\frac{3}{4}\left(\frac{q}{L\,\sigma_{id}}\right)\frac{1}{\tilde{d}_1\tilde{d}_2^2} \leq 1, \tag{5.2.8}$$

$$\frac{3\sqrt{3}}{4}\left(\frac{q}{L\,\sigma_{id}}\right)\frac{1}{\tilde{d}_1\tilde{d}_2} \leq 1, \tag{5.2.9}$$

$$\frac{5\cdot 125}{16}\left(\frac{q}{E\,L}\right)\frac{1}{\tilde{d}_1\tilde{d}_2^3} \leq 1. \tag{5.2.10}$$

It is useful to define two dimensionless non-negative parameters, collected in $\tilde{\boldsymbol{b}}$, which completely characterize the fixed variables of the problem:

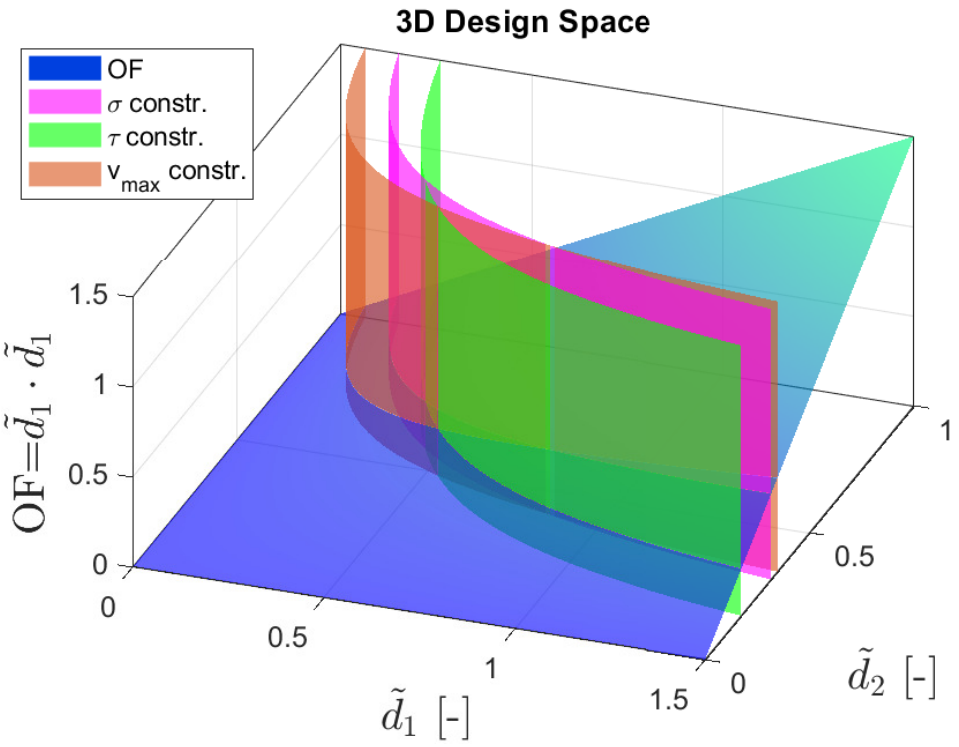
$$\psi_\sigma = \frac{q}{L\,\sigma_{id}}, \tag{5.2.11}$$

$$\psi_E = \frac{q}{E\,L}. \tag{5.2.12}$$

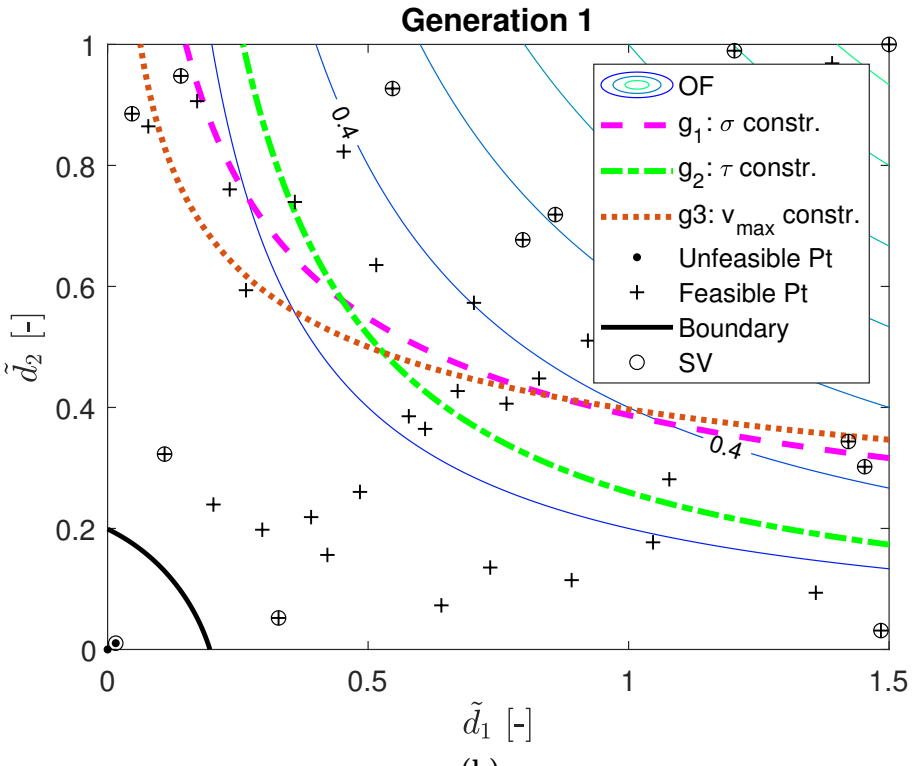
Finally, the following dimensionless version of the problem is formulated (5.2.7)

$$\begin{aligned} \min \quad & \tilde{f}(\tilde{d}_1,\tilde{d}_2) = \tilde{d}_1\tilde{d}_2, \\ \text{s.t.} \quad & \tilde{d}_1 > 0,\ \tilde{d}_2 > 0, \\ & \frac{3}{4}\psi_\sigma\frac{1}{\tilde{d}_1\tilde{d}_2^2} \leq 1, \\ & \frac{3\sqrt{3}}{4}\psi_\sigma\frac{1}{\tilde{d}_1\tilde{d}_2} \leq 1, \\ & \frac{5\cdot 125}{16}\psi_E\frac{1}{\tilde{d}_1\tilde{d}_2^3} \leq 1. \end{aligned} \tag{5.2.13}$$

To solve this problem, the PSO-SVM is adopted with the piecewise linear decreasing relax constraint function with a user coefficient $\lambda$ for standard deviation of the unfeasible points fixed to $\lambda = 0.05$. For academic purposes, in order to graphically analyze the behaviour of the constraint handling, the dimensionless parameters are fixed as $\psi_\sigma = \psi_E = 0.2$, looking for a optimal solution in design domain for the dimensionless design variables as $0 \leq \tilde{d}_1 \leq 1.5$ and $0 \leq \tilde{d}_2 \leq 1$. In this way, with this particular choice of the dimensionless parameters the constraints intersect each other creating discontinuous non-linear boundary of the feasible region. The population size is always 50 individuals and the maximum iterations are 100. After 50 runs the results shown a quite great variability of the design variables but always at the same objective function value. This fact enlightens the presence of a front of possible optimal solutions. In fact, as showed in the graphical representations in Figures 5.9-5.10-5.11, for this specific choice of $\tilde{\boldsymbol{b}}$, only two constraints are active and there exists a region of optimal solutions on the $\tau$ constraint boundary line. In the following, the comparison table with PSO-SVM, PSO-Penalty and GA shows the

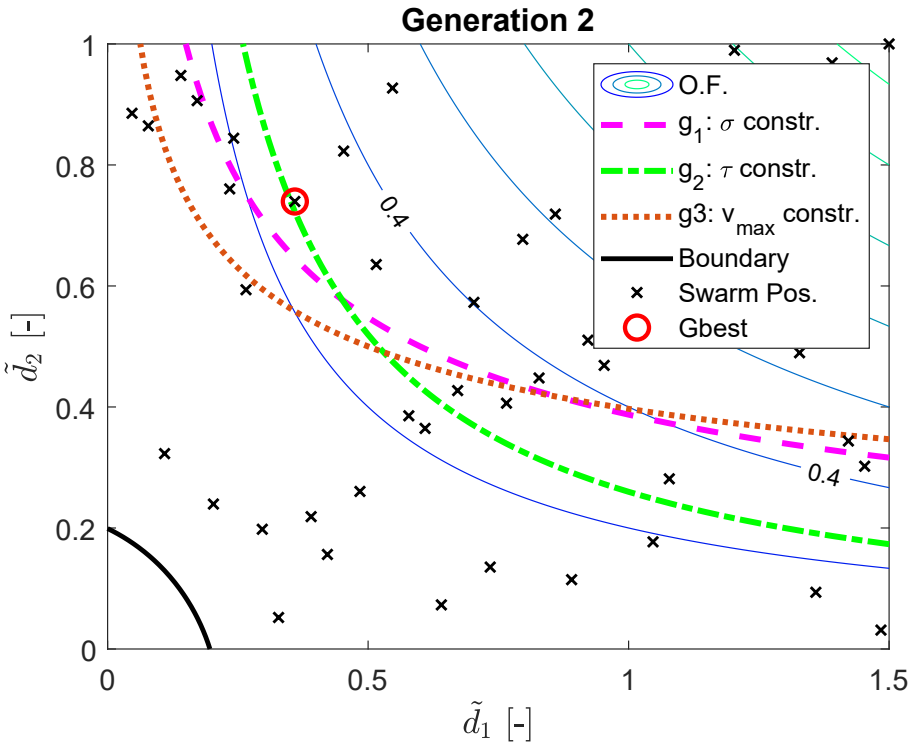



(a)

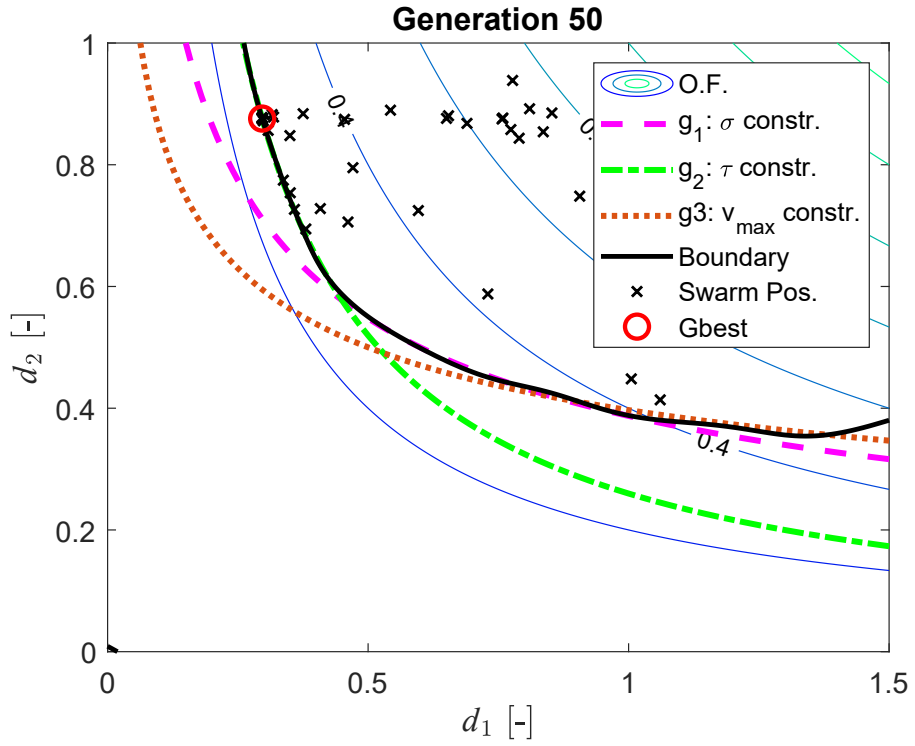



(b)

**Figure 5.9:** Structural example 1: simply supported beam, case piecewise linear decreasing relax function;
(a) Three-dimensional graph of simply supported beam problem design space; (b) Generation 1.

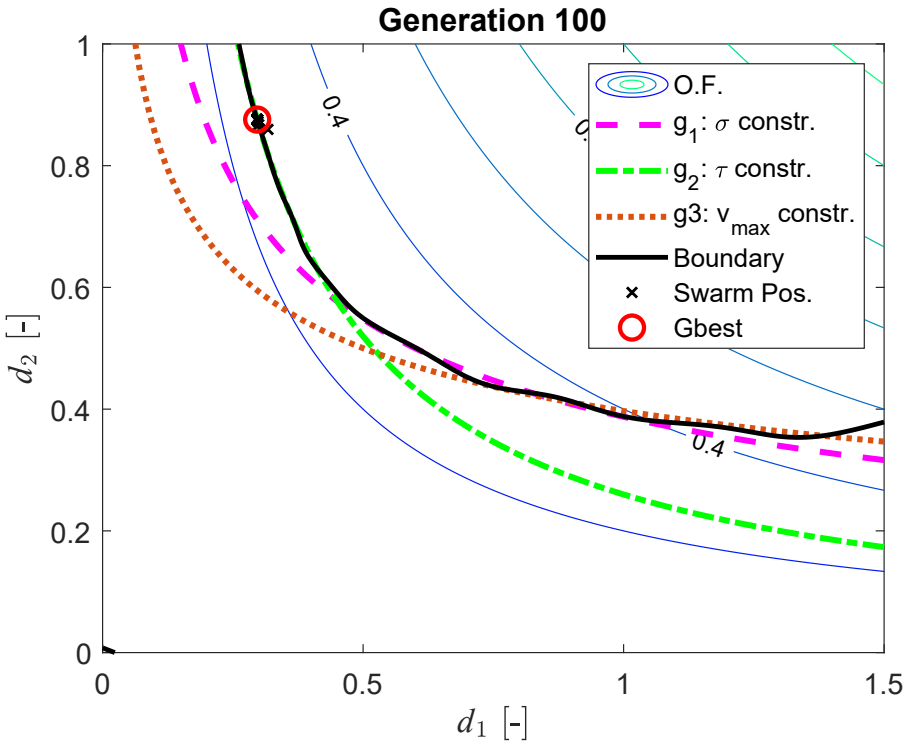



(a)

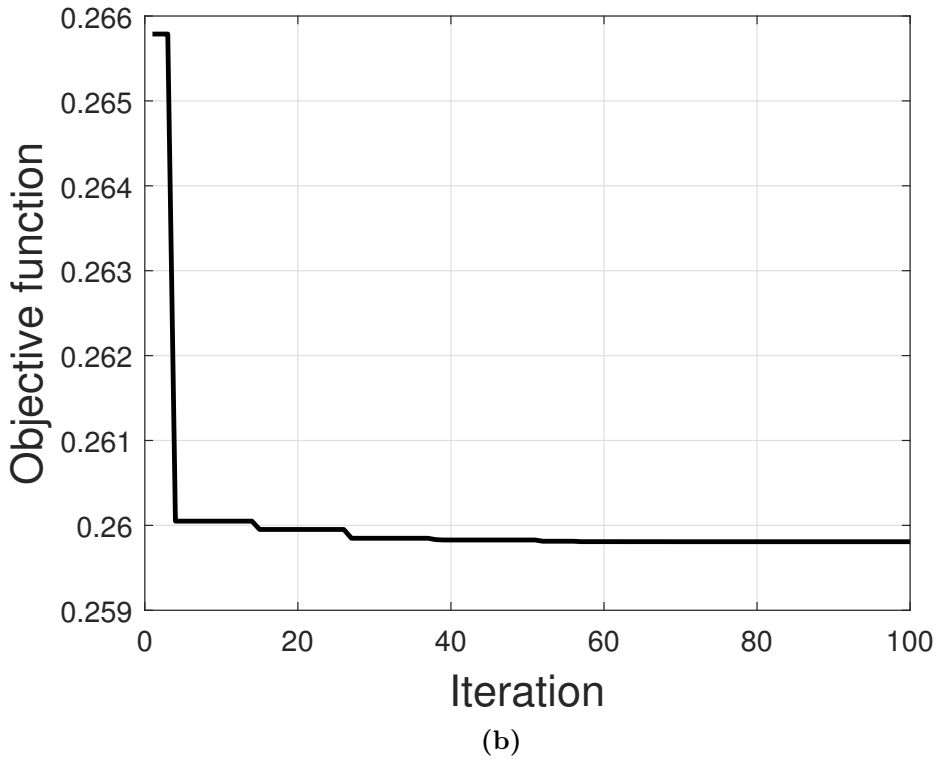



(b)

**Figure 5.10:** Structural example 1: simply supported beam, case piecewise linear decreasing relax function;
(a) Generation 2; (b) Generation 50.




(a)




(b)

**Figure 5.11:** Structural example 1: simply supported beam, case piecewise linear decreasing relax function;
(a) Generation 100; (b) objective function history.

mean value and the standard deviation of the best objective function value obtained after 50 runs.

In order to find the entire front of the all optimal possible solutions for this specific problem, it is necessary to find the all the pairs $(\tilde{d}_{1,opt}, \tilde{d}_{2,opt})$ posing the objective function as $f(\tilde{d}_1, \tilde{d}_2) = \tilde{d}_1 \cdot \tilde{d}_2 \approx 0.2598$. As one can see in Figures 5.9-5.10-5.11, this optimal front corresponds to a part of the $\tau$ constraint (5.2.9) posed as an equality. Referring to the $\tilde{d}_1$ optimal possible values, the optimum front is upper bounded from the $\sigma$ constraint (5.2.8) posed as an equality and lower bounded from the box search space limits. Then, to calculate the optimal upper bound $\tilde{d}_{1,\text{opt,UB}}$ it is necessary to calculate the intersection between the two aforementioned constraints

$$\begin{aligned} &\frac{3}{4}\psi_\sigma \frac{1}{\tilde{d}_1 \tilde{d}_2^2} = 1, \\ &\frac{3\sqrt{3}}{4}\psi_\sigma \frac{1}{\tilde{d}_1 \tilde{d}_2} = 1, \\ &\sqrt{\frac{3}{4}\psi_\sigma \frac{1}{\tilde{d}_1}} = \frac{3\sqrt{3}}{4}\psi_\sigma \frac{1}{\tilde{d}_1} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} &\tilde{d}_2 = \sqrt{\frac{3}{4}\psi_\sigma \frac{1}{\tilde{d}_1}}, \\ &\tilde{d}_2 = \frac{3\sqrt{3}}{4}\psi_\sigma \frac{1}{\tilde{d}_1}, \end{aligned} \quad \Longrightarrow \quad \tilde{d}_{1,\text{opt,UB}} = \frac{9}{4}\psi_\sigma = 0.45. \tag{5.2.14}$$

As before, to calculate the optimal lower bound $\tilde{d}_{1,\text{opt,LB}}$ it is necessary to calculate the intersection between the equality $\tau$ constraint and the horizontal line $\tilde{d}_2 = 1$, obtaining

$$\begin{aligned} &\frac{3\sqrt{3}}{4}\psi_\sigma \frac{1}{\tilde{d}_1 \tilde{d}_2} = 1, \\ &\tilde{d}_2 = 1, \end{aligned} \quad \Longrightarrow \quad \tilde{d}_{1,\text{opt,LB}} = \frac{3\sqrt{3}}{4}\psi_\sigma = 0.2598. \tag{5.2.15}$$

Finally, considering (5.2.14) and (5.2.15), it is possible to obtain all the optimal pairs ($\tilde{d}_{1,opt}$,
$\tilde{d}_{2,opt}$) using the following equation:

$$\tilde{d}_{2,\text{opt}} = \frac{0.2598}{\tilde{d}_{1,\text{opt}}}, \quad \text{with} \quad 0.2598 \leq \tilde{d}_{1,\text{opt}} \leq 0.45. \tag{5.2.16}$$

Since the algorithm works with dimensionless parameters, in order to find the physical dimensions of the optimized cross-section it is sufficient to multiply the

**Table 5.4:** Structural Example 1, results from PSO-SVM piecewise linear decreasing relax constraints, PSO-Penalty and GA.

| | PSO-SVM | PSO-Penalty | GA |
|---|---|---|---|
| $\mu_{OF}$ | 0.2598076 | 0.2598076 | 0.2610035 |
| $\sigma_{OF}$ | 2.04e-09 | 5.61e-17 | 2.58e-03 |
| Best OF | 0.2598076 | 0.259808 | 0.2598076 |
| Worst OF | 0.2598076 | 0.259808 | 0.2664507 |

obtained values $(\tilde{d}_{1,opt}, \tilde{d}_{2,opt})$ by $L$.

To show a technical possibly application coming from this simple example, a only concrete beam with span length $L = 3$ m is now considered. Disregarding for the moment the self-weight load, the $q$ load set to 15 kN/m represents only a live load. The concrete modulus is set to $E = 25$ GPa and the Von Mises ideal stress is related to the tensile stress of concrete set to $\sigma_{id} = 3$ MPa. Considering the box search space as $0 \leq b \leq 40$ cm and $0 \leq h \leq 45$ cm, the algorithm found the minimum weight respecting the constraints with $b = 16.67$ cm and $h = 42$ cm. Rounding-off these values, the self-weight associated to a concrete beam with $b = 18$ cm and and $h = 45$ cm is equal to:

$$G = \gamma_{concrete} \cdot b \cdot h = 24 \cdot 0.18 \cdot 0.45 = 1.944 \text{ kN/m}.$$

For sake of simplicity, adding $G$ to $q$ a new load equal to 16.944 kN/m which takes into account also the self-weight is defined. Launching again the algorithm, new optimal exact dimensions are now obtained: $b = 18.83$ cm and $h = 45$ cm. Rounding-up the exact solution, a new self-weight equal to $G = 2.16$ kN/m is coming from a section with $b = 20$ cm and $h = 45$ cm. Now the convergence is reached because the new optimal exact solution is $b = 19.07$ cm and $h = 45$ cm. Finally, the optimal cross-section for this concrete beam which minimize the self-weight is given by $b = 20$ cm and $h = 45$ cm.

### 5.2.2 Structural Example 2: Optimization of a Warren Truss Beam

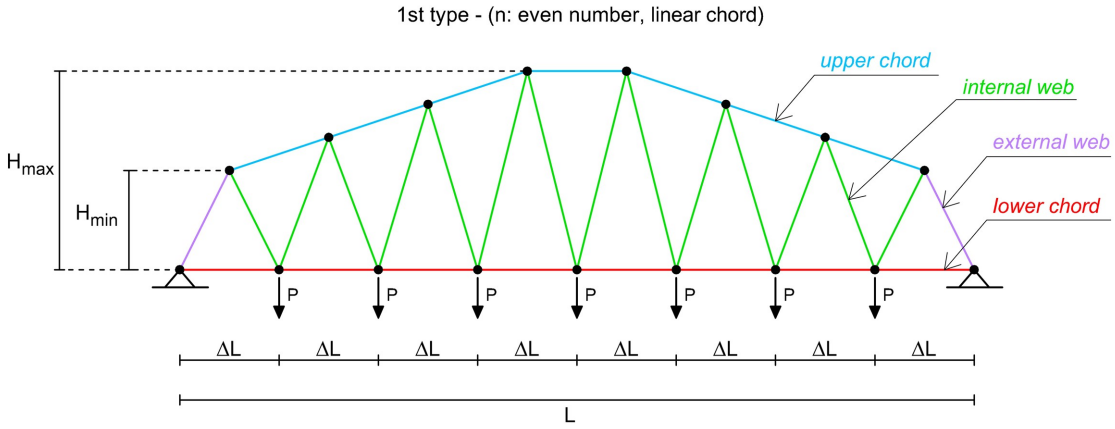



**Figure 5.12:** Problem formulation: simply supported truss Warren beam.

The second structural example comes from [23]. In that work the weight optimization of an in-plane Warren truss simply supported beam, depicted in Figure 5.12, is performed with Differential Evolutionary Algorithm (DEA). The steel profile used for truss members is a square hollow core section, as shown in Figure 5.13. This kind of profile ensures good stability against buckling and it represents a good solution for this type of structure because of its high strength-to-weight ratio [23]. On the other hand, joint connections are usually welded and so, in order to reduce the total cost, it is important to reduce the size of sections to be welded.

Regarding to the size optimization problem, as shown in Figure 5.13, this kind of sections are completely described by only two independent design variables: the outer dimension of the cross section $B$ and the thickness of the webs $s$. In this problem the truss has $m$ members belonging to four different type of cross sections as reported in Figure 5.12: lower chord $(B_1, s_1)$, upper chord $(B_2, s_2)$, internal webs $(B_3, s_3)$ and external webs $(B_4, s_4)$. To perform the shape optimization further two design variables are considered: vertical height of the external webs $H_{min}$ and the maximum height $H_{max}$. The design vector is therefore defined as

$$\boldsymbol{x} = (B_1, B_2, B_3, B_4, s_1, s_2, s_3, s_4, H_{min}, H_{max}), \tag{5.2.17}$$

and the box search space domain $\Omega$ is defined by:

$$\begin{aligned} 60 \leq & B_i \leq 360 \text{ mm}, \\ 4 \leq & s_i \leq 30 \text{ mm}, \\ 50 \leq & H \leq L \text{ mm}, \end{aligned} \tag{5.2.18}$$

where $i = 1, 2, 3$ and 4 and $L$ is the total span lenght. Considering the maximum value of thickness $s_{max} = 30$ mm, the minimum value of dimension $B$ cannot be assumed less than $2s_{max}$ due to geometric limits. The objective function is represented

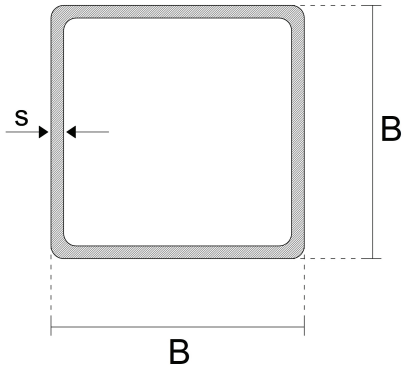



**Figure 5.13:** Square hollow core tubular section design variables.

by the total weight of the structure (Marano et al., 2006):

$$W(\boldsymbol{x}) = \sum_{i=1}^{m} \rho_i l_i A_i, \tag{5.2.19}$$

where $\rho_i = 7.85$ t/$m^3$ is the steel density supposed equal for all members, $l_i$ is the length and $A_i$ is the cross section of the $i$-th member. The structural steel used in this example is a S275 and the modulus of elasticity of the steel is 210 GPa. Regarding topology optimization, in [23] for fixed length $L = 20$ m, the optimal number of bays in which divide the lower chord is 20. The external load is as a uniformly distributed load $q = 100$ kN/m applied on the lower chord acting as point loads in the nodes of the truss. The constrains are represented by the strength verifications about tensile stress (without any holes) (5.2.20), compression stress (5.2.21) and buckling instability (5.2.22) according to Eurocode 3 (EN 1993-1 2005 and EN 1993-2 2006). Despite from the Eurocode $\gamma_{M0} = 1$ and $\gamma_{M1} = 1.1$ for bridges are recommended, to be more safe the partial safety factors are set both equal to $\gamma_{M_0} = \gamma_{M_1} = 1.1$.

$$\frac{N_{Ed}}{N_{t,Rd}} \leq 1, \text{ where } N_{t,Rd} = \frac{A f_y}{\gamma_{M_0}}, \tag{5.2.20}$$

$$\begin{aligned} \frac{N_{Ed}}{N_{c,Rd}} \leq 1, \text{ where for classes 1,2,3 } N_{c,Rd} &= \frac{A f_y}{\gamma_{M_0}}, \\ \text{while for class 4 } N_{c,Rd} &= \frac{A_{eff} f_y}{\gamma_{M_1}}, \end{aligned} \tag{5.2.21}$$

$$\begin{aligned} \frac{N_{Ed}}{N_{b,Rd}} \leq 1, \text{ where for classes 1,2,3 } N_{c,Rd} &= \chi \frac{A f_y}{\gamma_{M_1}}, \\ \text{while for class 4 } N_{c,Rd} &= \chi \frac{A_{eff} f_y}{\gamma_{M_1}}. \end{aligned} \tag{5.2.22}$$

Another constraint to satisfy is the maximum deflection which is usually set to $u_{lim} = L/500$ for bridges like that. In order to make a comparison with the results of [23], for academic reasons, there is no distinction of the load combination for the strength verifications and for the deformability checks. The verification equations

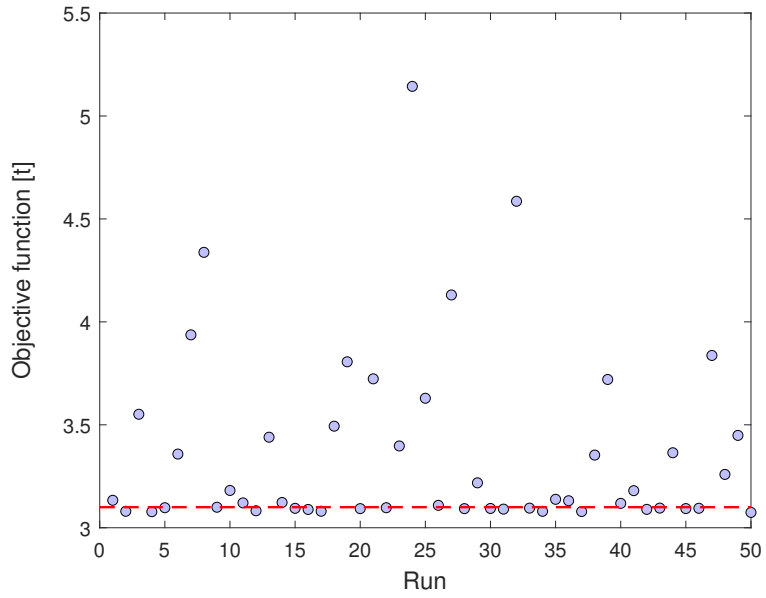



**Figure 5.14:** Structural example 2: Warren Truss. Results from 50 times run PSO-SVM.

not need to deeper examination because this is beyond the scope of the present document. Therefore, the optimization problem statement is the following [23]: *Find* $\boldsymbol{x} \in \Omega$ *such that*

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = W(\boldsymbol{x}), \\ \text{s.t.} \quad & \frac{N_{Ed}}{N_{t,Rd}} \leq 1, \\ & \frac{N_{Ed}}{N_{c,Rd}} \leq 1, \\ & \frac{N_{Ed}}{N_{b,Rd}} \leq 1, \\ & u_{max} \leq u_{lim}. \end{aligned} \tag{5.2.23}$$

In order to solve (5.2.23), the FEM structural analysis was performed in the Matlab® CALFEM and PSO-SVM was adopted for the optimization process. In the PSO-SVM a population size of 100 individuals is set with $k_{max} = 200$ iterations and a constant relax function with user parameter $\lambda = 3$ applied to standard deviation of unfeasible points. It is performed 50 times runs and the best-obtained solutions are collected in Figure 5.14. As one can see in Figure 5.14, due to the complexity of the problem, sometimes the algorithm not reach the optimum and stack in a local minimum. As shown in the graph, the optimum solution is around 3.1 t, then it is possible to cut the graph considering only the 21 runs over the total 50 which are characterized by a best OF solution lower than 3.1 t (dashed line). In this way, the possible outliers are excluded and now it is possible to perform the post-processing searching for the real best solution.

Considering the above-mentioned solutions, the obtained results showed in Table 5.5 has a quite large standard deviation in terms of design variables but very low in terms of objective function. This means that it is possible to find many

**Table 5.5:** Structural Example 2, Mean values $\mu$ and standard deviations $\sigma$ of best results from 21 solution over 50 runs of PSO-SVM with OF less than 3.1 t; Last three columns: Best exact solution, Trivial rounded-up solution and Refined industrial solution.

| [mm] | $\mu$ Exact Sol. | $\sigma$ Exact Sol. | Best Sol. | Trivial Industrial Sol. | Best Industrial Sol. |
|---|---|---|---|---|---|
| $B_1$ | 72.5 | 15.7 | 94.2 | 95 | 95 |
| $B_2$ | 190.2 | 70.2 | 128.1 | 130 | 105 |
| $B_3$ | 128.9 | 1.4 | 128.8 | 130 | 130 |
| $B_4$ | 211.7 | 107.9 | 132.1 | 135 | 110 |
| $s_1$ | 6.0 | 1.5 | 4 | 4 | 4 |
| $s_2$ | 14.7 | 7.4 | 18.8 | 20 | 26 |
| $s_3$ | 4.0 | 0.03 | 4 | 4 | 4 |
| $s_4$ | 14.0 | 8.4 | 14.7 | 15 | 19 |
| $H_{min}$ | 399.4 | 21.5 | 410.5 | 410 | 410 |
| $H_{max}$ | 4064.7 | 113.5 | 4145.0 | 4145 | 4145 |
| $OF$ [t] | 3.0898 | 0.0071 | 3.074 | 3.189 | 3.092 |

combinations of design variables which is giving always the almost same objective function. The best design value of the 50 runs, also reported in table 5.5, is taken into consideration. It is possible to compare this latter objective function (3.074 t) with the optimal exact solution given by the original DEA code output [23]. The DEA optimal solution was characterized by the weight of 2.95 t so this is in the right order of magnitude. The comparison of the design variables is based only on general observation in accordance with the literature, as affirmed in [23]. In fact, for instance, it is expected that, mainly due to instability problems, upper chord and external diagonals would be bigger than the lower chord and internal diagonals. Finally, the best exact solution may be trivially rounded-up to get an industrialized more realistic design. The new design variables are reported in 5.5 and as one can see this solution is more conservative and it leads to an increased total weight (3.189 t).

If one want to find a more accurate industrial solution it is necessary to perform a more accurate analysis of the obtained results. Since the topology optimization was already taken into account in [23], one have to remember that in the design variables the algorithm is performing the size optimization and the shape optimization. This latter is regarding to the definition of the $H_{min}$ and $H_{max}$ values. Considering only the above-mentioned 21 solutions and the standard deviations of $H_{min}$ and $H_{max}$, it is possible to assume that the rounded-up values of the best exact solution can represent a good result for shape optimal parameters values:

$$\begin{aligned} H_{min} &\approx 410 \text{ mm}, \\ H_{max} &\approx 4145 \text{ mm}. \end{aligned} \tag{5.2.24}$$

Once solved the shape optimization, regarding to the size optimization firstly the

**Table 5.6:** Structural Example 2, Best exact solution cross section

| $[\text{mm}, \text{mm}^2]$ | $(B; s)_{exact}$ | Requested Area | $(B; s)^{best}_{industrial}$ | Provided Area |
|---|---|---|---|---|
| Lower Chord | (94.2 ; 4) | 1442.46 | (95 ; 4) | 1456 |
| Upper Chord | (128.1 ; 18.8) | 8206.18 | (105 ; 26) | 8216 |
| Internal Webs | (128.8 ; 4) | 1996.81 | (130 ; 4) | 2016 |
| External Webs | (132.1 ; 14.7) | 6907.21 | (110 ; 19) | 6916 |

best exact solution as the optimal one. One have to remember that $B_i$ and $s_i$ were chosen as design parameters because of their independence, but in the optimization process, they are connected. In fact, in both objective function evaluation and constraints evaluation, these two parameters are combined into the resisting cross section value. It is possible to obtain almost the same value of cross section with different combinations of the design parameters. In particular, one can refer to the optimal exact solution in terms of resisting cross sections which represent the best solution in terms of both strength verification and minimization of the weight. As one can check, for the best exact solution, the section class of all members is 1, but other optimal solutions within the 21 considered are characterized by class 4 profile. In this case, to get the strength verification satisfied, it is necessary to refer to the resisting effective area. Usually in the design when it is possible it is preferred to avoid class 4 profiles and the best condition is to find an optimal solution with class 1 profile. Therefore, the best industrial solution which respects all the constraint is given by all the pairs $(B_i, s_i)$ with $i = 1, 2, 3$ and 4 which gives class 1 profiles and which gives the minimum value of area greater or equal than the effective areas requested by the best solution in Table 5.6. This procedure allows us to find the best solution which respects the strength verification only. For the instability verification, it is necessary to take into account also the second moment of inertia which condition the Euler's critical load $N_{cr}$ and consequently the dimensionless slenderness $\overline{\lambda}$ which influence the reduction factor $\chi$. Fixing the thickness $s_i$ to discrete values (rounded with 1 mm of precision), starting from the best-found solution and making an iterative discrete research to find the $B_i$ (rounded with 5 mm of precision) respectful of our above-mentioned design rule, the best optimal industrial solution is found and reported in the last column of Table 5.5. As one can check, the minimum cross area of internal webs could be assured by the pair $(B_3, s_3) = (105\ ;\ 5)$ mm but, due to instability problems, it is necessary to take into account the inertia and choose a profile that ensures both strength and instability verifications.
The best industrial structure is verified to all strength and instability constraints. Making a comparison between the two last columns of the Table 5.5, also the trivial rounded-up solution represents a good optimal solution in terms of objective function. In fact, minimizing the weight is important but the total cost is also affected by other aspects e.g welding and detailing, labour cost, etc. So, the solution obtained by trivial rounded-up the exact one it can be considered an acceptable optimal result. In Figure 5.15 the undeformed and the deformed shape are depicted. It is possible

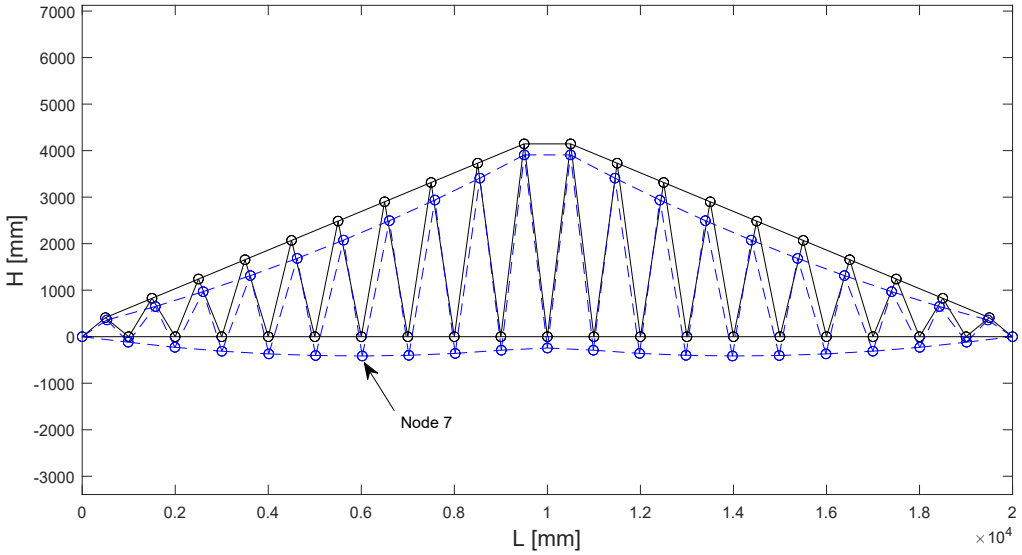



**Figure 5.15:** Optimal Warren Truss. Black solid line: undeformed shape; blue dashed line: deformed shape.

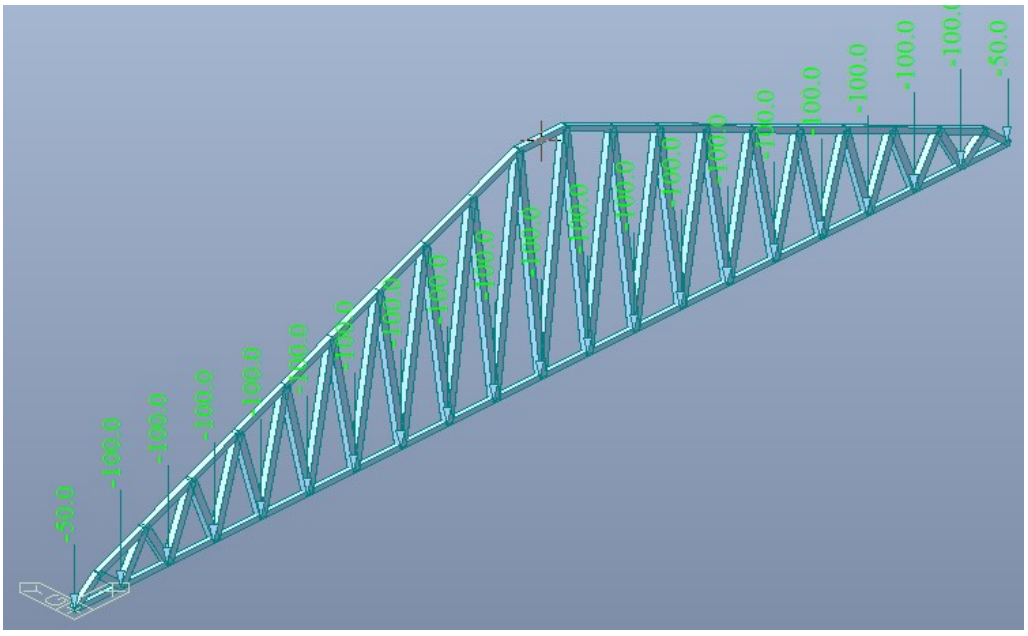


**Figure 5.16:** Model of the warren truss beam in Midas Gen®.

to appreciate which the node 7 and, due to symmetry, the node 15 are the nodes that undergo the most deflection $u_{max} = 39.8$ mm however it is respectfully of the service limit $L/500 = 40$ mm.

In order to assess the validity of the results of the optimization process, the warren truss beam is modelled with FEM professional software MIDAS Gen®. The assessment is made not only in terms of axial force for each member, but in particular in terms of performance ratio. This latter represents an efficiency percentage of the usage of the steel and is given by the strength ratio between the demand and the

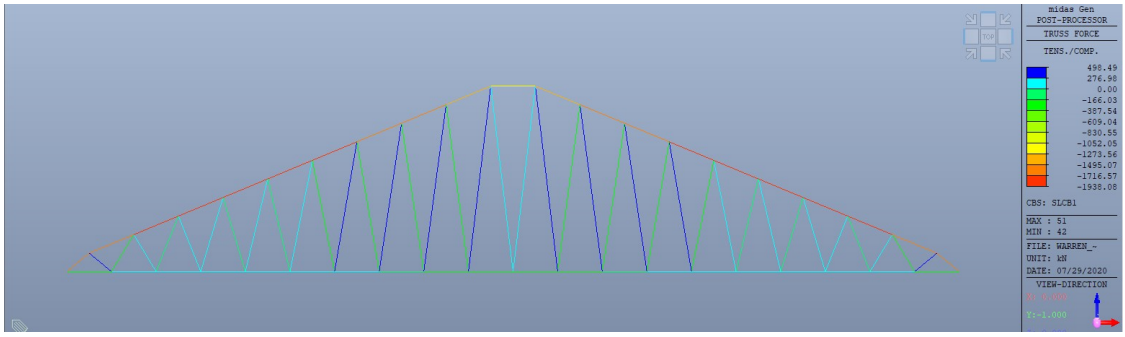



**Figure 5.17:** Planar view of the warren truss beam on Midas Gen® with the axial force values.

capacity. The simply supported warren truss is modelled through truss elements in order to guarantee pure axial behaviour of each member. The section properties of the trivial best industrial results from Table 5.5 are assigned to each element. The adopted steel is always a S275 and the uniformly distributed load $q = 100$ kN/m acting on the lower chord is reconducted as a hanged load directly applied at lower chord nodes acting as concentrated forces, as depicted in Figure 5.16. As already remarked, in order to get results which are directly comparable with the Matlab code and with DEA code from [23] and for the sake of simplicity, none load combination is considered. The aim is to demonstrate that the proposed algorithm provides comparable results with DEA code which is used as a benchmark and not making a perfect design completely respectful of the current codes. In order to take into account this latter issue, it is sufficient to consider the correct the load combination at ULS for the strength verification and SLS for the maximum allowable displacement.

As reported in Figure 5.17 and in Table 5.7, the results obtained from the Matlab code are equal to the FEM software. The overall behaviour shows that the upper chord is entirely compressed whereas the lower chord is entirely tensed. The internal webs are alternatively compressed and tensed as usual for truss beam of this typology. Calculating the strength ratio, it is worth noting that a performance ratio between 75% to 98% is obtained for the members. These remarkable results show the importance of the optimization process during the design phase which can strongly support the decision process of the designer.

**Table 5.7:** Comparison between Midas model and Matlab axial force elements

| Midas Element | Matlab Element | $\|N_{Ed}\|$ | $\|N_{Rd}\|$ |
|---|---|---|---|
| 1 | 1 | 350.47 | 364.00 |
| 2 | 2 | 166.18 | 364.00 |
| 3 | 3 | 255.42 | 364.00 |
| 4 | 4 | 238.70 | 364.00 |
| 5 | 5 | 179.70 | 364.00 |
| 6 | 6 | 99.54 | 364.00 |
| 7 | 7 | 7.27 | 364.00 |
| 8 | 8 | 92.60 | 364.00 |
| 9 | 9 | 197.58 | 364.00 |
| 10 | 10 | 306.16 | 364.00 |
| 11 | 11 | 306.16 | 364.00 |
| 12 | 12 | 197.58 | 364.00 |
| 13 | 13 | 92.60 | 364.00 |
| 14 | 14 | 7.27 | 364.00 |
| 15 | 15 | 99.54 | 364.00 |
| 16 | 16 | 179.70 | 364.00 |
| 17 | 17 | 238.70 | 364.00 |
| 18 | 18 | 255.42 | 364.00 |
| 19 | 19 | 166.18 | 364.00 |
| 80 | 20 | 350.47 | 364.00 |
| 40 | 21 | 1692.70 | 2200.00 |
| 41 | 22 | 1918.80 | 2200.00 |
| 42 | 23 | 1939.50 | 2200.00 |
| 43 | 24 | 1892.10 | 2200.00 |
| 44 | 25 | 1813.70 | 2200.00 |
| 45 | 26 | 1718.60 | 2200.00 |
| 46 | 27 | 1613.50 | 2200.00 |
| 47 | 28 | 1501.80 | 2200.00 |
| 48 | 29 | 1385.70 | 2200.00 |
| 49 | 30 | 1228.10 | 2200.00 |
| 78 | 31 | 1385.70 | 2200.00 |
| 77 | 32 | 1501.80 | 2200.00 |
| 76 | 33 | 1613.50 | 2200.00 |
| 75 | 34 | 1718.60 | 2200.00 |
| 74 | 35 | 1813.70 | 2200.00 |
| 73 | 36 | 1892.10 | 2200.00 |
| 72 | 37 | 1939.50 | 2200.00 |
| 71 | 38 | 1918.80 | 2200.00 |
| 70 | 39 | 1692.70 | 2200.00 |
| 21 | 40 | 498.81 | 504.00 |
| 23 | 41 | 150.24 | 504.00 |
| 25 | 42 | 20.80 | 504.00 |
| 27 | 43 | 66.77 | 504.00 |
| 29 | 44 | 139.27 | 504.00 |
| 31 | 45 | 204.59 | 504.00 |
| 33 | 46 | 265.90 | 504.00 |
| 35 | 47 | 324.76 | 504.00 |
| 37 | 48 | 382.00 | 504.00 |
| 39 | 49 | 50.75 | 504.00 |
| 68 | 50 | 482.83 | 504.00 |
| 66 | 51 | 425.57 | 504.00 |
| 64 | 52 | 366.75 | 504.00 |
| 62 | 53 | 305.56 | 504.00 |
| 60 | 54 | 240.59 | 504.00 |
| 58 | 55 | 169.06 | 504.00 |
| 56 | 56 | 84.69 | 504.00 |
| 54 | 57 | 30.40 | 504.00 |
| 52 | 58 | 252.62 | 504.00 |
| 22 | 59 | 252.62 | 504.00 |
| 24 | 60 | 30.40 | 504.00 |
| 26 | 61 | 84.69 | 504.00 |
| 28 | 62 | 169.06 | 504.00 |
| 30 | 63 | 240.59 | 504.00 |
| 32 | 64 | 305.56 | 504.00 |
| 34 | 65 | 366.75 | 504.00 |
| 36 | 66 | 425.57 | 504.00 |
| 38 | 67 | 482.83 | 504.00 |
| 69 | 68 | 50.75 | 504.00 |
| 67 | 69 | 382.00 | 504.00 |
| 65 | 70 | 324.76 | 504.00 |
| 63 | 71 | 265.90 | 504.00 |
| 61 | 72 | 204.59 | 504.00 |
| 59 | 73 | 139.27 | 504.00 |
| 57 | 74 | 66.77 | 504.00 |
| 55 | 75 | 20.80 | 504.00 |
| 53 | 76 | 150.24 | 504.00 |
| 51 | 77 | 498.81 | 504.00 |
| 20 | 78 | 1523.00 | 1800.00 |
| 50 | 79 | 1523.00 | 1800.00 |

# Chapter 6

# Conclusions

First and foremost, it is important to point out that the present Thesis does not intend to be excessively pretentious or even exhaustive about the vast world of the Optimization. Nevertheless, this work could be rather considered as a valid, albeit brief, introduction for those who want to approach and explore for the first time the optimization field from its origin up to the current development with AI and machine learning, with special regard to the structural optimization sub-field. The present dissertation even offers some solid literature references for deepened insights in the subject. Furthermore, the present Thesis could be treated, in turn, as a reference for future researches and further developments in order to contribute enriching the scientific knowledge.

After an introductory part concerning the classical mathematical approaches which are mainly gradient-based, some of the most important meta-heuristic optimization procedures have been presented highlighting not only the advantages but also the drawbacks of these modern search procedures. In the central part of the Thesis, the attention is mainly focused on the particle swarm optimization (PSO) algorithm showing that this subject is still under development in order to improve and make more robust the search process. Therefore, in this Thesis, a new constraint handling approach have been introduced combining the machine learning method of Support Vector Machine (SVM) with classical Newtonian PSO.

The PSO-SVM represents a new valid alternative not-penalty method to solve constrained optimization problems. Its main advantage respect to the most used nowadays penalty approach is represented by the generality of the machine learning SVM algorithm. Since it depends intrinsically on the inner product of the data, it is more adaptive even with discontinuous and non-linear boundary of the feasible region in the design space. In order to improve the behaviour of the proposed algorithm to deal with very sharp and narrow feasible region, a relax constraint function has also been implemented. From the computational point of view, the trade-off was found by adopting a limited population size and by using an incremental boundary update. It would be also possible sampling a huge initial random data and leaving the boundary fixed during the generations, however, this procedure does not lead to a good result in terms of objective function.

Finally, the two numerical benchmark examples demonstrate the convergence

of the new method in comparison with another penalty approach and with a GA. The last two examples highlight the adaptability of this new method even into the structural optimization field. In particular, in the warren truss beam problem, the optimization algorithm provided a numerical exact solution which can be easily industrialized by the designer with a trivial rounding-off without jeopardizing the optimization process. Although the warren truss beam example is performed under simplified assumption in order to make comparisons with DEA code from [23], from a technical point of view, the new optimization algorithm becomes a really useful and powerful support for the designer during the design and the decision process. It is important to stress that working with meta-heuristic algorithms always involves the definition of the value of many arbitrary parameters. Starting from literature suggestions for these values, it is always strongly suggested to perform a fine-tuning of some of these parameters, even with a trial-and-error approach for each specific problem in order to find the best optimal results in terms of objective function convergence, computational effort and elaboration time.

# Appendix A

# Test Functions Constrained Problems

The following mathematical problems were tested for the proposed PSO-SVM algorithm.

1. The following problem is taken by [54] and it is called Sickle function.

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = (x_1 - 20)^3 + (x_2 - 10)^3 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0 \\ & g_2(\boldsymbol{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 82.81 \geq 0, \end{aligned}$$

   where the search space is defined as $0 \leq x_1 \leq 10$ and $14 \leq x_2 \leq 15.5$. The global optimum is located at $\boldsymbol{x}^* = [14.095, 0.84296]$ where $f(\boldsymbol{x}) = -6961.8139$.

2. The following problem is taken from [67] and it is a multi-variable problem with five design variables and six constraints.

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0, \\ & g_2(\boldsymbol{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0, \\ & g_3(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0, \\ & g_4(\boldsymbol{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0, \\ & g_5(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0, \\ & g_6(\boldsymbol{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0, \end{aligned}$$

   where the search space is defined as $78 \leq x_1 \leq 102$ and $33 \leq x_2 \leq 45$ and $27 \leq x_3, x_4, x_5 \leq 45$. The optimum is located at $\boldsymbol{x}^* = [78, 33, 29.995256025682, 45, 36.775812905788]$ where $f(\boldsymbol{x}) = -30,665.539$.

# Bibliography


[1] Begambre O., Laier J.E., "*A hybrid Particle Swarm Optimization – Simplex algorithm (PSOS) for structural damage identification*", Advances in Engineering Software, 40(9), 883-891, 2009.

[2] Beyer, H., Schwefel, H., "*Evolution strategies – A comprehensive introduction*". Natural Computing 1, 3–52 (2002). https://doi.org/10.1023/A:1015059928466

[3] Bishop, C. M. "*Pattern Recognition and Machine Learning*", Springer Science+Business Media (2006) pp. 3, 291-296, 325-336

[4] Bloomfield M.W., Herencia J.E., Weaver P.M., "*Analysis and benchmarking of meta-heuristic techniques for lay-up optimization*", Computers & Structures, 88(5-6), 272-282, 2010.

[5] Canuto, C. and Tabacco, A., "*Mathematical Analysis I*", Springer Science+Business Media, Second Edition, Springer International Publishing Switzerland, 2015.

[6] Carpinteri, A. "*Scienza delle costruzioni 2*", Vol. 2, Terza Edizione, Pitagora Editrice, Bologna (1995)

[7] Carvalho, É.C.R., Bernardino, H.S., Hallak, P.H. et al. "*An adaptive penalty scheme to solve constrained structural optimization problems by a Craziness based Particle Swarm Optimization*". Optim Eng 18, 693–722 (2017). https://doi.org/10.1007/s11081-016-9344-z

[8] Chen T.-Y., Chi T.-M., "*On the improvements of the particle swarm optimization algorithm*", Advances in Engineering Software, 41(2), 229-239, 2010.

[9] Christensen, P.W. et al. "*An Introduction to Structural Optimization*", Springer Netherlands, Springer Science+Business Media B.V. (2009) pp. 44

[10] Coello C.A.C., "*Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art*", Computer Methods in Applied Mechanics and Engineering, 191(11-12), 1245-1287, 2002.

[11] Coello Coello, C. A., Lamont, G. B., van Veldhuizen, D. A., "*Evolutionary Algorithms for Solving Multi-Objective Problems*", Springer Science+Business Media, Second Edition, New York, USA, 2007.

[12] Coello, C. A. C., "*Self-adaptive penalties for GA-based optimization,*" Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 573-580 Vol. 1, doi: https://doi.org/10.1109/CEC.1999.781984.

[13] Congiu, E., "*Parametric design and optimization of arched trusses under vertical and horizontal multi-load cases*", PhD in Civil Engineering and Architecture, Università degli Studi di Cagliari (2020) http://hdl.handle.net/11584/290547

[14] Cortes C. and Vapnik V. "*Support-Vector Networks*", Kluwer Academic Publishers, Boston (1995)

[15] Cristianini N. and Schölkopf B. "*Support Vector Machines and Kernel Methods*", AI Magazine Volume 23 Number 3 (2002)

[16] de Castro Lemonge, A.C., Duarte, G.R., da Fonseca, L.G. "*An algorithm inspired by bee colonies coupled to an adaptive penalty method for truss structural optimization problems*". J Braz. Soc. Mech. Sci. Eng. 41, 126 (2019). https://doi.org/10.1007/s40430-019-1629-7

[17] Deb K., "*An efficient constraint handling method for genetic algorithms*", Computer Methods in Applied Mechanics and Engineering, 186(2-4), 311–338, 2000.

[18] Dhiman, G., Kumar, V. "*Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications*", Adv. Eng. Softw. 114 (2017): 48-70.

[19] Emmerich, M.T.M., Deutz, A.H. "*A tutorial on multiobjective optimization: fundamentals and evolutionary methods*". Springer, Natural Computing Journal 17, 585–609 (2018). https://doi.org/10.1007/s11047-018-9685-y

[20] Ertel, W., "*Introduction to Artificial Intelligence*", Springer Nature, Springer International Publishing AG (2017)

[21] F. Mota, V. Almeida, E. F. Wanner and G. Moreira, "*Hybrid PSO Algorithm with Iterated Local Search Operator for Equality Constraints Problems*", 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, 2018, pp. 1-6.

[22] Falcone, R., Lima, C., Martinelli, E. "*Soft computing techniques in structural and earthquake engineering: a literature review*", Engineering Structures, Vol. 207, (2020), https://doi.org/10.1016/j.engstruct.2020.110269

[23] Fiore, A., Marano, G.C., Greco, R. et al. "*Structural optimization of hollow-section steel trusses by differential evolution algorithm*". Int J Steel Struct 16, 411–423 (2016). https://doi.org/10.1007/s13296-016-6013-1

[24] Gandomi, A.H., Yang, X., Alavi, A.H., "*Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems*". Engineering with Computers 29, 17–35 (2013). https://doi.org/10.1007/s00366-011-0241-y

[25] Ge H.-W., Liang Y.-C., Marchese M., "*A modified particle swarm optimization-based dynamic recurrent neural network for identifying and controlling nonlinear systems*", Computers and Structures, 85(21-22), 1611-1622, 2007.

[26] Gholizadeh S., Salajegheh E., "*Optimal design of structures subjected to time history loading by swarm intelligence and an advanced metamodel*", Computer Methods in Applied Mechanics and Engineering, 198(37-40), 2936-2949, 2009.

[27] Gunantara, N. "*A review of multi-objective optimization: Methods and its applications*". Cogent Engineering, 5(1), 1502242. (2018) https://doi.org/10.1080/23311916.2018.1502242

[28] Haddad, O.B., Solgi, M., Loáiciga, H.A., "*Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*". John Wiley & Sons, USA (2017).

[29] Hasançebi O., Çarbaş S., Doğan E., Erdal F., Saka M.P., "*Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures*", Computers and Structures, 87(5-6), 284-302, 2009.

[30] Jordehi, A.R. "*A review on constraint handling strategies in particle swarm optimisation*". Neural Comput & Applic 26, 1265–1275 (2015). https://doi.org/10.1007/s00521-014-1808-5

[31] Kalivarapu V., Foo J.-L., Winer E., "*Synchronous parallelization of Particle Swarm Optimization with digital pheromones*", Advances in Engineering Software, 40(10), 975-985, 2009.

[32] Kar, R., Mandal, D., Mondal, S., Ghoshal, S.P. "*Craziness based Particle Swarm Optimization algorithm for FIR band stop filter design*", Swarm and Evolutionary Computation, Vol. 7, 58-64, (2012) https://doi.org/10.1016/j.swevo.2012.05.002

[33] Kaveh A., Talatahari S., "*Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures*", Computers and Structures, 87(5-6), 267-283, 2009.

[34] Kennedy J., Eberhart R.C., "*Particle swarm optimization*", in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, Vol. 4, 1942-1948, 1995.

[35] Kennedy J., Eberhart R.C., "*Swarm Intelligence*", Morgan Kaufmann, San Francisco, CA, 2001.

[36] Koziel S., Michalewicz Z., "*Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization*", Evolutionary Computation, 7(1), 19-44, 1999.

[37] Li B., Xiao R.Y., "*The Particle Swarm Optimization Algorithm: How to Select the Number of Iteration*", Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing IIHMSP 2007, Kaohsiung, Vol. 2, 191-196, 2007.

[38] LI Hong-shuang, LÜ Zhen-zhou, YUE Zhu-feng "*Support Vector Machine for structural reliability analysis*", Applied Mathematics and Mechanics English Ed. (2006)

[39] Li L.J., Huang Z.B., Liu F., "*A heuristic particle swarm optimization method for truss structures with discrete variables*", Computers and Structures, 87(7-8), 435-443, 2009.

[40] Liang J.J., Qin A.K., Suganthan P.N., Baskar S., "*Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions*", IEEE Transactions on Evolutionary Computation, 10(3), 281-295, 2006.

[41] M. Kohler, L. Forero, M. Vellasco, R. Tanscheit and M. A. Pacheco, "*PSO+: A nonlinear constraints-handling particle swarm optimization*", 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 2518-2523.

[42] Mahamad Nabab Alam, "*Codes in MATLAB for Particle Swarm Optimization* Research Scholar, ResearchGate, March 2016, DOI: 10.13140/RG.2.1.1078.7608

[43] Martí, R., Pardalos, P. M., Resende, M. G. "*Handbook of Heuristics*", Springer Nature Switzerland (2018)

[44] Medina, A. J. R. et al. "*A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers*". IJCCI 2009: 152-159, Proceedings Of The International Joint Conference On Computational Intelligence, (2009)

[45] Mezura-Montes E. (Ed.), "*Constraint-Handling in Evolutionary Optimization*", Studies in Computational Intelligence Series Vol. 198, Springer, 2009.

[46] Mezura-Montes E., Coello C.A.C., "*A simple multimembered evolution strategy to solve constrained optimization problems*", IEEE Transactions on Evolutionary Computation, 9(1), 1-17, 2005.

[47] Michalewicz Z., Fogel D.B., "*How to Solve It: Modern Heuristics*, Springer, Berlin, 2000.

[48] MirjaliliJin, S., Dong , J.S. "*Multi-Objective Optimization using Artificial Intelligence Techniques*". SpringerBriefs in Computational Intelligence, Switzerland (2020). https://doi.org/10.1007/s11047-018-9685-y

[49] Monti G., Quaranta G., Marano G.C., "*Genetic-Algorithm-Based Strategies for Dynamic Identification of Nonlinear Systems with Noise-Corrupted Response*", ASCE Journal of Computing in Civil Engineering, 24(2), 173-187, 2010.

[50] Monti G., Quaranta G., Marano G.C., "*Genetic-Algorithm-Based Strategies for Dynamic Identification of Nonlinear Systems with Noise-Corrupted Response*", ASCE Journal of Computing in Civil Engineering, 24(2), 173-187, 2010.

[51] Murphy, K. P. "*Machine Learning: A Probabilistic Perspective*", Massachusetts Institute of Technology Press (2012) pp. 2-3, 479-481, 488, 496-501

[52] Omkar S.N., Mudigere D., Naik G.N., Gopalakrishnan S., "*Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures*", Computers and Structures, 86(1-2), 1-14, 2008.

[53] Omkar, S.N. et al. "*Artificial immune system for multi-objective design optimization of composite structures*", Engineering Applications of Artificial Intelligence, Vol. 21-8, (2008) Pag. 1416-1429, https://doi.org/10.1016/j.engappai.2008.01.002

[54] P. A. Simionescu, D. G. Beale and G. V. Dozier, "*Constrained optimization problem solving using estimation of distribution algorithms*", Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Portland, OR, USA, 2004, pp. 296-302 Vol.1, doi: 10.1109/CEC.2004.1330870.

[55] Parsopoulos K.E., Vrahatis M.N. "*Particle Swarm Optimization Method for Constrained Optimization Problems*".In Proceedings of the Euro-International Symposium on Computational Intelligence (2002)

[56] Parsopoulos K.E., Vrahatis M.N. "*Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems*". In: Wang L., Chen K., Ong Y.S. (eds) Advances in Natural Computation. ICNC 2005. Lecture Notes in Computer Science, vol 3612. Springer, Berlin, Heidelberg. (2005) https://doi.org/10.1007/11539902_71

[57] Perez R.E., Behdinan K., "*Particle swarm approach for structural design optimization*", Computers and Structures, 85(19-20), 1579-1588, 2007.

[58] Plevris V., Papadrakakis M., "*A hybrid Particle Swarm-Gradient algorithm for global structural optimization*", Computer-Aided Civil and Infrastructure Engineering, 2010 https://doi.org/10.1111/j.1467-8667.2010.00664.x.

[59] Plevris, V. *Innovative Computational Techniques for the Optimum Structural Design Considering Uncertainties*, National Technical University of Athens, Athens (2009)

[60] Plevris, V., Mitropoulou, C.C., Lagaros, N.D. "*Structural Seismic Design Optimization and Earthquake Engineering: Formulations and Applications*". IGI Global, First edition (2012)

[61] Praveen C., Duvigneau R., "*Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design*", Computer Methods in Applied Mechanics and Engineering, 198(9-12), 1087-1096, 2009.

[62] Quaranta, G., Lacarbonara, W. & Masri, S.F. "*A review on computational intelligence for identification of nonlinear dynamical systems*", Nonlinear Dyn 99, 1709–1761 (2020). https://doi.org/10.1007/s11071-019-05430-7

[63] Rao A.R.M., Sivasubramanian K., "*Multi-objective optimal design of fuzzy logic controller using a self configurable swarm intelligence algorithm*", Computers and Structures, 86(23-24), 2141-2154, 2008.

[64] Rao Singiresu S., *Engineering Optimization Theory and Practice*, John Wiley & Sons, Fifth Edition, USA (2019)

[65] Raschka, S., Mirjalili, V., "*Python machine learning*", II ed., Packt Publishing Ltd, Birmingham, UK (2017)

[66] Rothwell, A., "*Optimization Methods in Structural Design*", Solid Mechanics and Its Applications, Springer Nature, Springer International Publishing AG, Switzerland (2017)

[67] Runarsson, T. P., Yao, X. "*Search biases in constrained evolutionary optimization*", in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 35, no. 2, pp. 233-243, May 2005.

[68] Runarsson, T. P., Yao, X. "*Stochastic ranking for constrained evolutionary optimization*", IEEE Transactions on Evolutionary Computation, 4(3), 284-294, 2000.

[69] S. Das and P. N. Suganthan, "*Differential Evolution: A Survey of the State-of-the-Art*", in IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[70] Schmitt, B. I. "*Convergence Analysis for Particle Swarm Optimization*", FAU University Press, Erlangen, Nürnberg, Germany (2015)

[71] Sengupta, S.; Basak, S.; Peters, R.A., II. *Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives*, Mach. Learn. Knowl. Extr. 2019, 1, 157-191

[72] Seyedpoor S.M., Salajegheh J., Salajegheh E., Gholizadeh S., "*Optimum shape design of arch dams for earthquake loading using a fuzzy inference system and wavelet neural networks*", Engineering Optimization, 41(5), 473-493, 2009.

[73] Shi Y., Eberhart R.C., "*A modified particle swarm optimizer*", IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 69-73, 1998.

[74] Shi Y., Eberhart R.C., "*Empirical study of particle swarm optimization*", Congress on evolutionary computation, Washington D.C., Vol. 3, pp 1945–1950, 1999.

[75] Sonmez, M. "*Artificial Bee Colony algorithm for optimization of truss structures*", Applied Soft Computing, Vol. 11-2, (2011) https://doi.org/10.1016/j.asoc.2010.09.003

[76] Spillers, W.R. et al. "*Structural Optimization*", Springer US, Springer-Verlag US (2009) pp. 1-5

[77] Sun, J., Feng, B., Xu, W. "*A global search strategy of quantum-behaved particle swarm optimization*", IEEE Conference on Cybernetics and Intelligent Systems, 2004., Singapore, 2004, pp. 111-116 vol.1, https://doi.org/10.1109/ICCIS.2004.1460396

[78] Sun, J., Feng, B., Xu, W. "*Particle swarm optimization with particles having quantum behavior*", Proceedings of the 2004 Congress on Evolutionary Computation (IEEE

Cat. No.04TH8753), Portland, OR, USA, 2004, pp. 325-331 Vol.1, https://doi.org/10.1109/CEC.2004.1330875

[79] Vanderplaats, G.N. "*Very Large Scale Optimization*". National Aeronautics and Space Administration, NASA/CR-2002 211768, Vanderplaats Research and Development, Inc. Colorado Springs, CO, USA (2002)

[80] Vapnik, V. N. "*Statistical Learning Theory*", John Wiley & Sons, Inc. New York (1998) pp. 19-20, 24-25, 401-406, 421-422

[81] Wang Y., Cai Z., Zhou Y., Fan Z., "*Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique*", Structural and Multidisciplinary Optimization, 37(4), 395-413, 2009.

[82] Yang, XS., He, XS. "*Mathematical Foundations of Nature-Inspired Algorithms*", SpringerBriefs in Optimization, Springer Nature Switzerland (2019) https://doi.org/10.1007/978-3-030-16936-7

[83] Zhu, X., Gong, P. Zhao, Z., Zhang, C. "*Learning similarity metric with SVM*", The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, (2012) https://doi.org/10.1109/IJCNN.2012.6252829