Università
degli Studi
di Ferrara

Aix*Marseille
université
Socialement engagée

DOTTORATO DI RICERCA IN

"Scienze dell'Ingegneria"

Tesi in co-tutela con *Université d'Aix-Marseille*

CICLO XXXIV

COORDINATORE Prof. Stefano Trillo

Demonstrator of a procedure based on machine learning and use of natural language to promote breakthrough design concepts

Settore Scientifico Disciplinare ICAR/08

Dottorando

Dott. Gulinelli Pietro

Pietro Gulinelli

Tutore

Università di Ferrara

Prof. Rizzoni Raffaella

Raffaella Rizzoni

co – Tutore

Université d'Aix-Marseille

Prof. Lebon Frédéric

NNT/NL : 0000AIXM0000/000ED000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
en cotutelle avec l'Université de Ferrare (Italie)
le 21 mars 2022 par

Pietro GULINELLI

Demonstrator of a procedure based on machine learning and use of natural language to promote breakthrough design concepts

Discipline
Sciences pour l'Ingénieur

Spécialité
Mécanique des solides

École doctorale
ED 353

Laboratoire/Partenaires de recherche

- Laboratoire de Mécanique et Acoustique, Aix-Marseille Université
- Department of Engineering, University of Ferrara (Italy)
- Société CES WORKS (France)

Composition du jury

Lionel ROUCOULES ENSAM	Rapporteur
Enrico SPACONE University of Chieti-Pescara	Rapporteur
Christian HOCHARD AMU	Examinateur
Lynda TAMINE-LECHANI IRIT	Examinatrice
Alberto CORIGLIANO Polytech of Milan	Examinateur
Alessandra APRILE University of Ferrara	Examinatrice
Frédéric LEBON AMU	Directeur de thèse
Raffaella RIZZONI University of Ferrara	Co-directrice de thèse

Affidavit

Je soussigné, Pietro GULINELLI, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de M. Frédéric LEBON (AMU) et Mme. Raffaella RIZZONI (Université de Ferrare, Italie), dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille, le 11 janvier 2022

Pietro Gulinelli

Affidavit

I, undersigned, Pietro GULINELLI, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of M. Frédéric LEBON (AMU) et Mme. Raffaella RIZZONI (University of Ferrara, Italy), in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the French national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, 11 janvier 2022

Pietro Gulinelli

Liste de publications et participation aux conférences

1) Liste des publications réalisées dans le cadre du projet de thèse :

    1. P. Gulinelli, A. Aprile, R. Rizzoni, Y-H. Grunevald, F. Lebon, Multiscale numerical analysis of TRM-reinforced masonry under diagonal compression tests, Buildings 2020; 10(11):196, DOI: 10.3390/buildings10110196
    2. P. Gulinelli, A. Aprile, R. Rizzoni, Y-H. Grunevald, F. Lebon, R. Lovisetto, S. Tralli, A FE model for TRM reinforced masonry walls with interface effects, Key Engineering Materials 2019; 817: 57-64, DOI: 10.4028/www.scientific.net/KEM.817.57

2) Participation aux conférences et écoles d'été au cours de la période de thèse :

    1. Conférence : The fifth international conference on soft computing & optimisation in civil, structural and environmental engineering (CIVIL-COMP-OPTI), Hannah Griffiths, 16-19 septembre 2019, Riva del Garda, Italie
    2. Conférence : 6th International seminar on mechanics of masonry structures strengthened with composite materials (MuRiCo6), Prof. Ing. Angelo di Tommaso, 26-28 juin 2019, Université de Bologna
    3. Ecole d'été : Mechanics and multiphysics modelling of intelligent materials and micro electro-mechanical systems AIAS, 17-20 juin 2019, Prof. Laura Vergani, Université de Ferrare
    4. Conférence : 5th French-Italian Meeting on Masonry (FIMM5), 31 mai – 1 juin 2018, Frédéric Lebon, Laboratoire de Mécanique et d'Acoustique, Marseille

Résumé

Lorsque l'on souhaite dérouler une approche disruptive de conception, l'objectif premier est de sortir du cadre des solutions connues. Différentes méthodes, comme le brainstorming, facilitent ce travail. Néanmoins elles se heurtent à différents obstacles :

- Le biais cognitif, qui conduit le concepteur à explorer des voies erronées et, plus grave encore, à en écarter d'autres au motif qu'elles semblent, dès le départ, vouées à l'échec.
- Le champ de connaissance, qui doit idéalement être le plus large et diversifié que possible.
- L'accessibilité à la connaissance. Dans un monde ou tout évolue très vite dans tous les domaines il est de plus en plus difficile de suivre l'ensemble des évolutions d'un domaine large.
- L'approche d'optimisation qui est vue comme une façon d'innover au moins de façon incrémentale, mais qui a un potentiel disruptive limité car elle est conditionnée par un espace de conception de dimension finie.

Face à un besoin nouveau, le cerveau y associe des principes, des concepts, pour générer des similitudes. Cette phase est très rapide et quasi inconsciente. Si son auteur la juge intéressante (si l'idée survie au biais cognitif) alors on cherche d'abords à la formuler avec des mots qui décrivent en général une architecture, des principes physiques ou des propriétés matériaux, avant d'être traduits par des modèles plus ou moins détaillés.

Ce processus d'utilisation de mots pour expliquer, partager, confronter, se reproduit par la suite pour faire avancer la réflexion. L'intérêt majeur résidant dans la capacité à expliquer à la fois des éléments très globaux et basics et des concepts très complexes et détaillés, en passant d'un domaine à l'autre instantanément.

L'objectif de cette thèse est de s'inspirer de la façon dont un cerveau fonctionne au moment où il génère une idée et ou son propriétaire va la formuler et de créer un premier démonstrateur logiciel basé sur des outils IA. Elle se base sur les travaux menés au sein du groupe CES WORKS qui est spécialisé en conception disruptive.

Mots clés : démonstrateur, conception disruptive, conceptual design, machine learning, langage naturel.

Abstract

The development of a breakthrough design approach firstly requires getting out from the space of known solutions. Different methods can facilitate this work, as brainstorming. Nonetheless, they are often subjected to different obstacles:

- The cognitive biases, not only leading designers to explore erroneous solution paths, but also to exclude other apparently unsuccessful paths since the beginning.
- The knowledge space, which must ideally be as large and varied as possible.
- The accessibility to knowledge. In a world where everything changes and develops very quickly, it is increasingly difficult to follow all the developments in all fields.
- The optimization approach, which is seen as a way to innovate at least incrementally, but which has a limited breakthrough potential, since it is conditioned by a design space of finite dimension.

When we face with a new need, our brain is able to associate it principles and concepts, generating similarities and thus new ideas. This phase is generally very rapid, and it is almost unconscious. If these ideas are considered interesting (i.e. if they survive cognitive biases), we will first try to formulate them with words, which generally describe an architecture, physical principles or material properties, before translate them into more or less detailed models.

This process of using words to explain, share and compare will be repeated again and again to move forward. The major interest lies in its ability of explaining very general and basic concepts as well as very detailed and complex ones, and of instantly moving from one field to another different one.

The objective of this thesis is to be inspired by the way a brain works when it generates an idea and when its owner formulates it, in order to create a first software demonstrator based on different machine learning tools (artificial intelligence). The current thesis is based on the work carried out within the group CES WORKS, which is specialized in breakthrough design.

Keywords : demonstrator, breakthrough design, conceptual design, machine learning, natural language

Acknowledgements

I would first like to thank the group CES WORKS, funder of the current thesis, and in particular its scientific expert Professor Yves-Henri Grunevald, who was my supervisor and whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and provided me with the tools I needed to choose the right direction.

I would also like to thank my thesis directors, Professors Frédéric Lebon and Raffaella Rizzoni, for their valuable guidance throughout my studies and for their patient support throughout my doctorate. You provided me with the tools for successfully completing my dissertation, and for bringing my work to a higher level.

I would like to acknowledge my colleagues from the group CES WORKS for the stimulating discussions as well as happy distractions to rest my mind outside of my research. I want to thank you for all the opportunities I was given to further my research.

In addition, I would like to thank my girlfriend, my family and my friends for their wise counsel and sympathetic ear. You are always there for me.

Contents

Abbreviations

Abbreviation	**Explanation**
AI	Artificial Intelligence
ANN	Artificial Neural Network
AS	Artificial Sample
BDH	Bad (or Biased) Design Habitude
BDP	Breakthrough Design Practice
BERT	Bidirectional Encoder Representation for Transformer
BOW	Bag Of Words
BPA	Back Propagation Algorithm
CA	Colliding Aspect
CAD	Computer Aided Design
CF	Clustering Feature
CL	Complementary Lexicon
COCA	Corpus Of Contemporary American English
COVID-19	Coronavirus Disease 2019
COV	Coefficient Of Variation
CV	Concept Variant
DC	DC
DS	Design Sample
DT	Decision Tree
EP	Engineering Parameter
EU	European Union
FEM	Finite Element Method
FDS	Fictitious Design Sample
GA	Genetic Algorithm
GEP	Generalized Engineering Parameter
KCA	Key Cognitive Ability
LTM	Long Term Memory
LTR	Left-To-Right
ML	Machine Learning
NLP	Natural Language Processing
OECD	Organisation for Economic Co-operation and Development
PAA	Principle of the Adopted Approach
PDE	Product Design Engineering
R&D	Research and Development
RF	Random Forest
SM	Sensory Memory
SP	Solution Principle
STM	Short Term Memory
TIPS	Theory of Inventive Problem Solving
TRIZ	Russian abbreviation of the Theory of Inventive Problem Solving
UK	United Kingdom
US	United States

Symbols

Symbols, superscripts and Greek letters used for artificial neural networks

Symbols

E^n	Sample error referred to the nth training sample
E_{ep}	Epoch error
E_{ep}^{t}	Epoch error referred to the training set
ΔE_{ep}^{t}	Percentage variation of E_{ep}^{t} between two different numbers of hidden neurons which were consecutively tested
E_{ep}^{v}	Epoch error referred to the validation set
ΔE_{ep}^{v}	Percentage variation of E_{ep}^{v} between two different numbers of hidden neurons which were consecutively tested
N^R	Number of right responses provided by the neural network in an epoch
N^{R_t}	Number of right responses (cf. N^R) referred to the training set
N^{R_v}	Number of right responses (cf. N^R) referred to the validation set
N^S	Number of samples
t_k	Target probability corresponding to the kth output neuron (cf. y_k)
w	Generic synaptic weight
Δw	Weight correction
x	Input of the generic neuron
x'	Neuron accumulated signal
y	Output of the generic neuron
y_k	Output of the kth neuron in the generic layer, it refers to output neurons when used with t_k

Superscripts

b	Bias neuron
h	Hidden layer
i	Input layer
o	Output layer

Greek letters

η	Learning rate
μ	Momentum
τ	Actual epoch

Symbols used in the K-means clustering method

C	Cluster
d	Euclidean distance
D	Dispersion in a partitioning
D^K	Dispersion corresponding to the partitioning P^K (cf. P^K)
$D^{K=1}$	Dispersion of the training samples around their centroid
K	Number of clusters
P	Partitioning (or partitioning level)
P^K	Partitioning (or partitioning level) of minimum dispersion composed by K clusters
R_D	Percentage reduction of D^K due to the increasing of K, with respect to $D^{K=1}$
ΔR_D	Increment of R_D

Symbols used for the genetic algorithm

D_{av}	Average dispersion in the population of solutions at a given generation
ΔD_{av}	Percentage variation of D_{av} between two consecutive generations
D_{max}	Maximum dispersion in the population of solutions at a given generation
D_{min}	Minimum dispersion in the population of solutions at a given generation
ΔD_{min}	Percentage variation of D_{min} between two consecutive generations
g	Number of genes per variable
G	Number of generations achieved
G_{max}	Maximum number of generations
N	Number of individuals (solutions) in the population at each generation
p_c	Cross-over probability
p_m	Mutation probability
r_{max}	Higher limit of the variable range
r_{min}	Lower limit of the variable range
Δr	Increment of distance between two values of the variable

Symbols used in the NLP methodology

N_w	Number of words in the generic training sentence
x_i	Relative position of the i^{th} word in the sentence
Δx	Length of a sentence portion (the sentence of length 1 being divided into N_w portions, cf. N_w)

Introduction

Our world is constantly changing. New products and processes are under continuous development by companies, which continually try to be a step ahead competitors. Most of the developed products early fail, as well as their associated business models, whereas only a very few part of them reach prominent market positions [1]. The extraordinary technologic development of the recent decades has further accelerated this process, and the crucial role of design, as an activity aimed to conceive and develop new product and process concepts, has more and more emerged. The great challenges of our days, as the climate changes or the recent generalised crisis due to the COVID-19 pandemic, which is heavily impacting the social and economic framework, more than ever highlight the need for breakthrough design solutions, able to produce great advantages in terms of technology, sustainability and social changes. The development of a breakthrough design approach firstly requires getting out from the space of known solutions [2]. Nonetheless, this is often subjected to different obstacles:

- The cognitive biases. Designers often tend to project following their own preferences, leading to consider a limited set of design options and to explore erroneous solution paths, thus excluding other apparently unsuccessful paths since the beginning. The design solutions which can be quickly and easily realized usually tend to be immediately chosen, even if they involve a poor outcome [3]. The reasons of this tendency are mainly two. From one side, the design practices which allow to enlarge the solution space and to improve the research of more interesting design options are often unknown or ignored. From the other side, the behaviour of the same designers, as of all humans, is affected by subconscious mental mechanisms, which are indeed heavily responsible for some dangerous "bad design habitudes". The influence of these mechanisms, called cognitive biases, can be easily observed when, for example, we refuse to do something by answering "no, it's too challenging", "no, I don't like it" or "no, I'm not used to do this". These statements seem perfectly normal, but they are very dangerous for our design thinking, since they preclude us the possibility to imagine and explore different and potentially more interesting things. Everybody is subjected to cognitive biases, even more open and smarter people [3–7].
- The knowledge space. A breakthrough design approach requires having a knowledge space as wide as possible and as various as possible since the beginning of the project, and the ability to generate analogies between different fields and disciplines, different physical principles, etc. [2]. Optimization, representing the most widespread design approach, is sometimes seen as a way to innovate, at least incrementally, but it cannot lead to disruptive solutions because it is conditioned by a fixed design space of finite dimension from the beginning [8]. Knowledge variety can be favoured by the multiplication of contributors with different specializations. However, their thinking and experiences are not always easy to be exploited and combined. Indeed, the ability to generate analogies depends on the openness of the same contributors, which is strongly linked to outer factors, as personality, cultural background and previous projects [1].
- The accessibility to knowledge. In a world where everything changes and develops very quickly, it is increasingly difficult to follow all the developments in all fields. We must therefore specialize, which however amplifies the previous obstacles. For the same reason, our brains are saturated with information, and we sometimes tend to "reinvent the wheel" even after having apparently explored the whole state of the art.

In order to overcome these obstacles, we tried to imagine the way our brain works when it is faced with a new need. When we are faced with a design need, our brain is able to associate it principles and concepts, generating analogies and thus new ideas. This phase is generally very rapid and almost unconscious. If these ideas are considered interesting (i.e. if they survive cognitive biases), we will try to formulate them by words (i.e. natural language), generally describing an architecture, physical principles, material properties, etc. These elements will be first set in the brain as words, before being translated into more or less detailed models. This original hypothesis is the base of this thesis work. The process of using natural language to explain, share and compare will be repeated again to move forward in the reflection. The major interest of natural language lies in its ability of explaining very general and basic concepts as well as very detailed and complex ones, of emphasizing changes and of instantly moving from one field to another different one [9]. The objective of this thesis is to be inspired by this process, using machine learning (artificial intelligence) to create a first original software demonstrator. This ambitious idea is based on the work carried out within the group CES WORKS, funder of the current thesis and specialized in breakthrough design.

The thesis is organized in three chapter. In chapter 1, we clarify our perspective on design, which represents our focus, and we present the purpose and the approach adopted in this work. In chapter 2, we discuss the reasons for adopting machine learning, and we then present the implemented machine learning methods. In chapter 3, we present our original natural language processing (NLP) methodology based on machine learning, and we then adapt this methodology to the design field to create our original demonstrator. The latter will be described in detail and finally tested with an original example.

1. Focus, Purpose and Approach

The current chapter is organized in three main sections which respectively introduce the focus (i.e. design), the purpose and the approach pursued in this thesis. The conceptual map in Fig. 1 presents the structure of the chapter, highlighting the fundamental topics on which the discussion is developed.

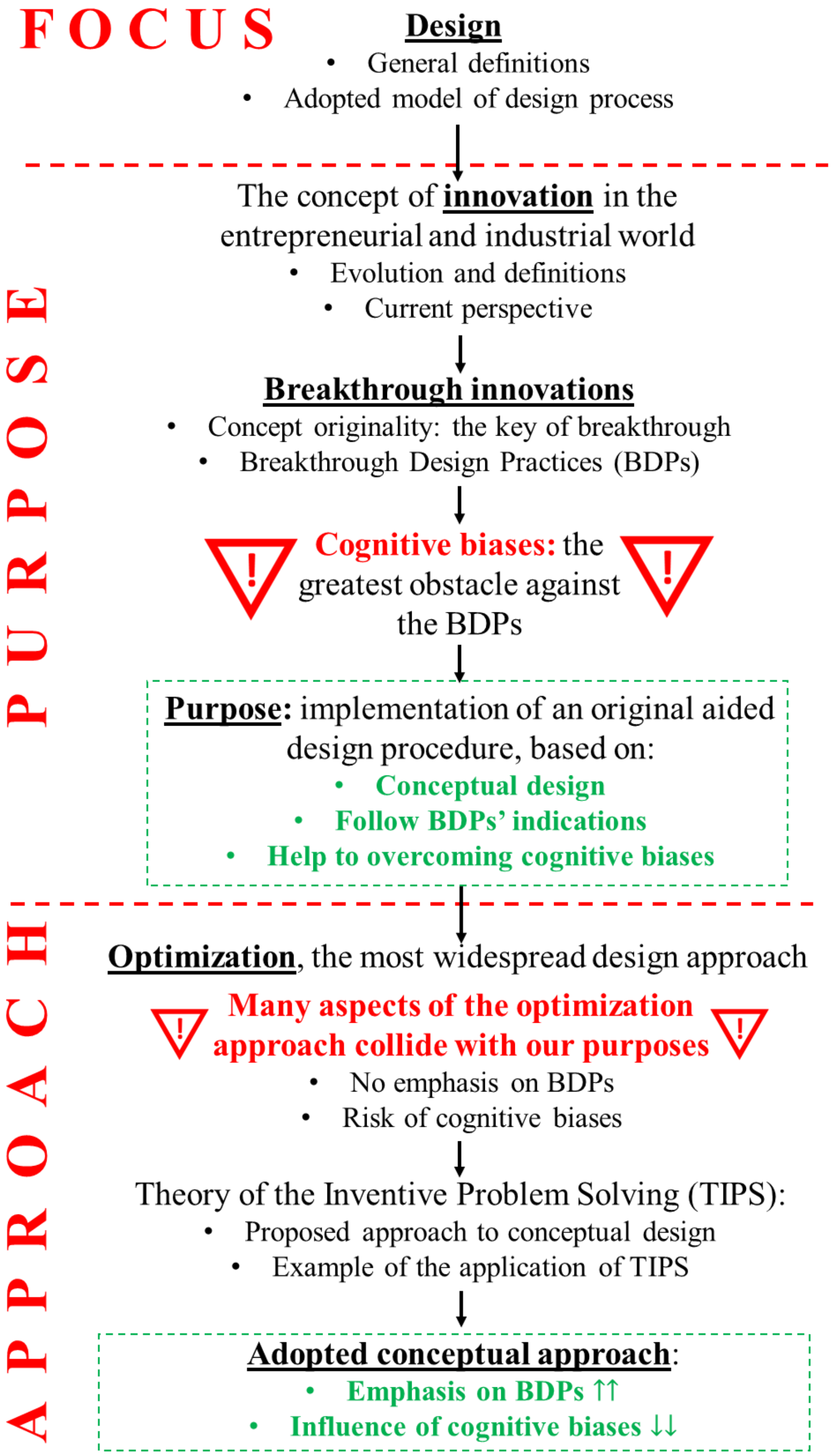

FOCUS
Design
• General definitions
• Adopted model of design process
PURPOSE
The concept of innovation in the entrepreneurial and industrial world
• Evolution and definitions
• Current perspective
Breakthrough innovations
• Concept originality: the key of breakthrough
• Breakthrough Design Practices (BDPs)
!
Cognitive biases: the greatest obstacle against the BDPs
!
Purpose: implementation of an original aided design procedure, based on:
• Conceptual design
• Follow BDPs' indications
• Help to overcoming cognitive biases
APPROACH
Optimization, the most widespread design approach
!
Many aspects of the optimization approach collide with our purposes
!
• No emphasis on BDPs
• Risk of cognitive biases
Theory of the Inventive Problem Solving (TIPS):
• Proposed approach to conceptual design
• Example of the application of TIPS
Adopted conceptual approach:
• Emphasis on BDPs ↑↑
• Influence of cognitive biases ↓↓

Fig. 1: Conceptual map of the chapter developed to explain the original path leading to the purpose and to approach of the thesis.

Since design represents the focus of the thesis work, we believe our perspective of this concept should be immediately clarified. Section 1.1 (“focus”, cf. Fig. 1) reports some important definitions about the roles of design as discipline and a well-established model of design process. The latter enables to highlight the crucial importance of the conceptual design phase. As argued in [10], design is very frequently linked to innovation by scholars, and this trend has particularly developed over the more recent decades. Notably, innovation is popularly intended as newness and associated to the technological progress, and it thus seems to more and more represent a fundamental component, or even the ultimate target, of the design activity [6,10]. We believe that a preliminary analysis of the innovation concept will help to better introduce our purpose. According to Fig. 1, a study of how innovation has been defined and perceived over the modern history (from 1960s to the recent decades) by industrials and entrepreneurs is firstly conducted in section 1.2 (“purpose”). This enables to outline our perspective of innovation in the current work and to focus on breakthrough innovation, popularly considered as the most valuable form of innovation. Notably, after identifying the distinctive element of breakthrough innovations, i.e. the concept originality, we introduce the fundamental design practices which can favour the birth of breakthrough innovations, denoted as BDPs (“Breakthrough” Design Practices, cf. Fig. 1). BDPs do not involve only technical aspects, but they also involve some important cognitive abilities which represent the source of the human creativity [2]. The study of some literature contributions (cf. [3–7]) allows to identify the greatest obstacle against BDPs: the cognitive biases (cf. Fig. 1), which are heavily responsible for “bad design habitudes”, denoted BDHs. The purpose of the thesis, expressed by the three fundamental principles reported in Fig. 1, is finally pointed out. But which design approach can be more suitable to reach our purpose? Section 1.3 (“approach”, cf. Fig. 1) starts by analysing the most widespread design approach, i.e. the optimization. The presence of many aspects of this approach colliding with our purposes reveals that optimization is not suitable to our needs. Notably, we demonstrate how optimization does not emphasize the BDP, and how the use of an aided design procedure based on this approach is strongly affected by designer’s cognitive biases. An alternative design approach is required. We thus proceed by studying the well-established conceptual design approach proposed by the Theory of Inventive Problem Solving (TIPS) [8,11]. Based on some important features of TIPS, we finally define the fundamental principles of the adopted design approach (cf. Fig. 1).

1.1. Focus

In this section, we want to briefly clarify our perspective of design, by outlining its roles as discipline, and by introducing a well-established model of design process. These aspects will be our reference for the concept of design during the rest of the thesis.

1.1.1. The roles of design

The word “design” is typically used to mainly indicate the general shaping of an object and the CAD tools used by engineers. However, identifying design with aesthetics or defining design as the process of collaborating with clients are two common practices [12,13]. The presence of all these meanings can result in a significant vagueness, which makes difficult to conceptualize the role of design as discipline [14]. Some efforts in this direction have however been made. In [15], Herbert famously defined design as “the process of changing existing states into preferred ones” A more recent and “less abstract” conceptualization is proposed by Hernandez in [10], who pointed out three main roles of design as discipline. These roles are reported in Table 1:

Table 1: The three main roles of design as discipline (cf. [10]).

The three main roles of design	
Design as …	Creative and generative thinking
	Research
	Differentiator

Contrary to the popularly diffused vision, these aspects, and especially the first role (cf. Table 1), highlight the centrality of the human mind in the design activity [10,15].

1.1.2. A well-established model of design process

Based on these ideas, Pahl et al. formulated in [16] a model of mechanical design process, which is schematically reported in Fig. 2according to [11,16]. This well-established model provides a sequential guideline consisting of four fundamental steps to support designers’ activities during product development (cf. Fig. 2). Fig. 2 clearly outlines the extremely iterative nature of the design process, in which any decision can determine a progress to the next phases or lead to reconsider the decisions made in each one of the previous steps. The first phase, i.e. task clarification, mainly concerns the collection of input data, the formulation of initial ideas and proposals to solve the given problem and the elaboration of the design specifications to satisfy.

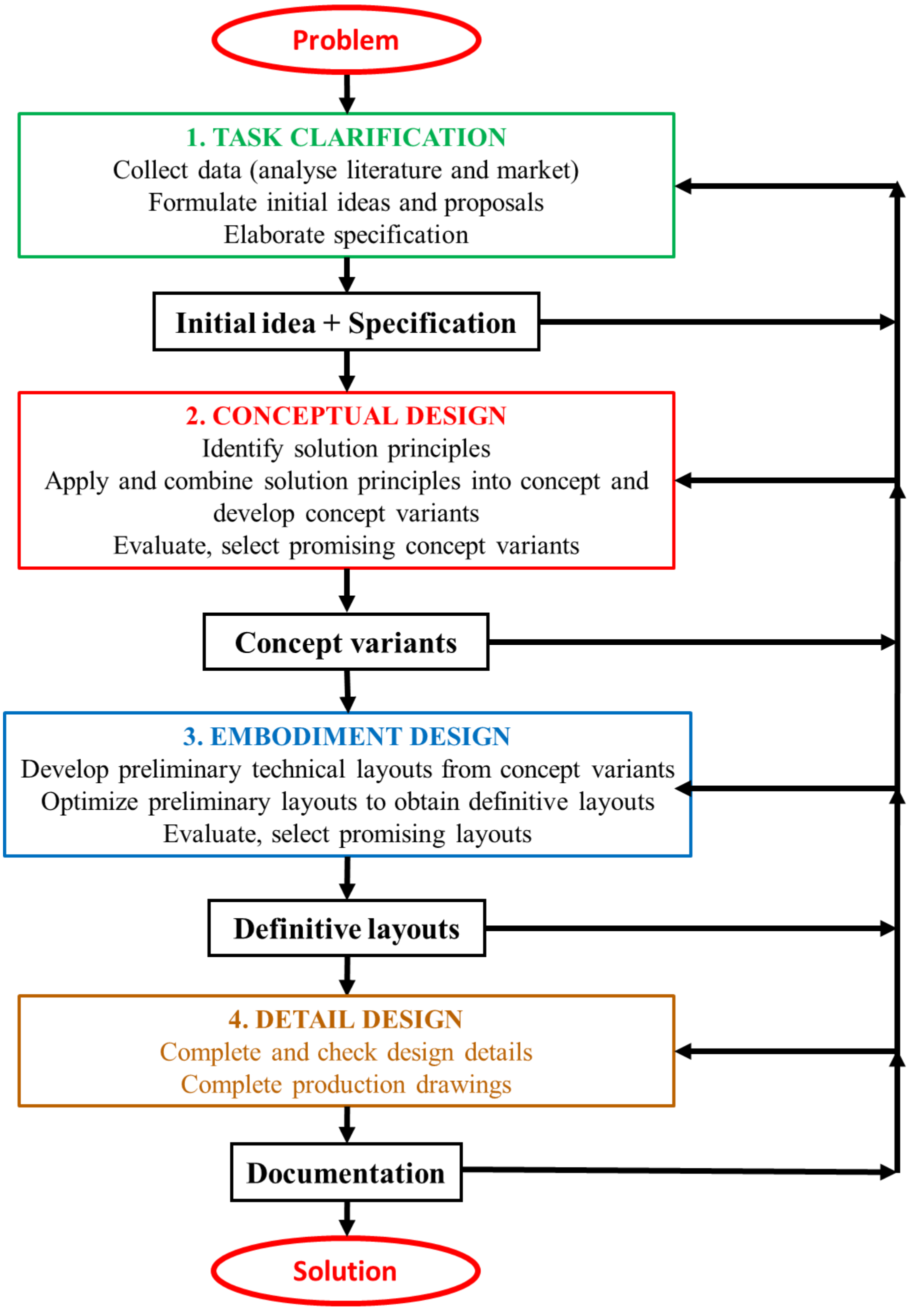

Problem
1. TASK CLARIFICATION
Collect data (analyse literature and market)
Formulate initial ideas and proposals
Elaborate specification
Initial idea + Specification
2. CONCEPTUAL DESIGN
Identify solution principles
Apply and combine solution principles into concept and develop concept variants
Evaluate, select promising concept variants
Concept variants
3. EMBODIMENT DESIGN
Develop preliminary technical layouts from concept variants
Optimize preliminary layouts to obtain definitive layouts
Evaluate, select promising layouts
Definitive layouts
4. DETAIL DESIGN
Complete and check design details
Complete production drawings
Documentation
Solution

Fig. 2: Model of mechanical design process (cf. [11,16]).

1.1.2.1. Conceptual design

The phase of conceptual design (second step, cf. Fig. 2) concerns the identification of the solution principles, i.e. the general strategies, to satisfy the required specifications. Solution principles are thus applied and combined to the initial idea, obtaining the concept, which represents an “evolution” of the initial idea towards the required specifications [11,16]. According to Fig. 2, the concept is further developed to obtain different concept variants. The example proposed in Fig. 3 helps to clarify the given definitions of solution principle, concept and variant (cf. Fig. 2). Notably, the purpose of this example is to schematically illustrate the conceptual design, without dwelling on the methods whereby solution principles are identified and applied to the initial idea.

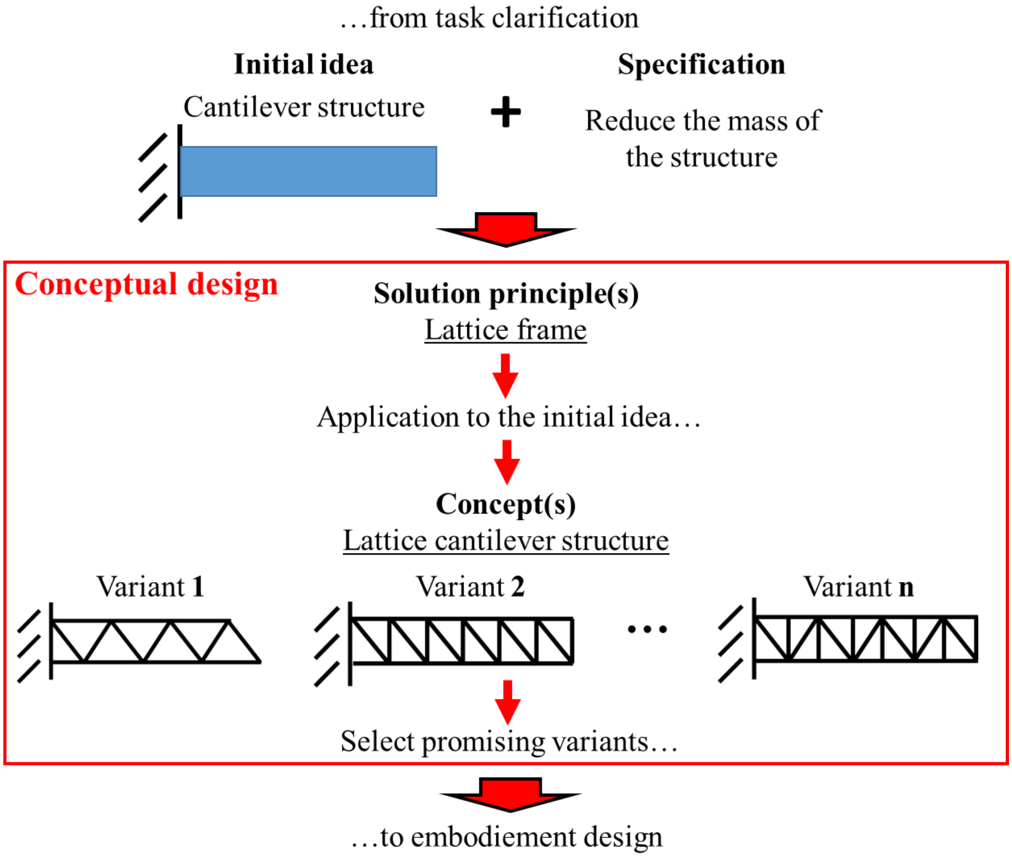


Fig. 3: Schematic example of conceptual design.

As shown in Fig. 3, the initial idea "cantilever structure" and the specification "reduce the mass…", arbitrarily chosen, constitute the input of conceptual design (cf. Fig. 2). First, the solution principle "lattice frame" is identified: the employment of a lattice frame represents a possible strategy to satisfy the required specification. Hence, the application of this principle to the initial idea leads to the concept "lattice cantilever beam", which can take shape into different concept variants. According to Fig. 3, the concept expresses the essential and constant characteristics of all variants (i.e. "being a lattice cantilever structure"), while each variant expresses a different declination of the concept (i.e. a different configuration of the lattice). The more solution principles are multiple and different, the more they can be combined into multiple and different concept variants. Theoretically, according to [11,16], there is no limit on how solution principles can be combined. At the end of conceptual design (cf. Fig. 2 and Fig. 3), concept variants are generally evaluated against economic criteria, and the most promising ones are selected. Some examples of real product concepts follow:

- Concept of Sony Walkman: portable device to bring along your own music. This concept involves the combination of one main principle, i.e. the portability of the music [2].
- Concept of joystick: ergonomic device to control your alter ego in a video game This concept involves the combination of two main principles, i.e. the criterion of ergonomicity and the control of an alter ego in a video game [17].

As argued in [11,16,18], the conceptual design has a considerable effect on important variables, as the cost, and it is thus known as the most crucial phase of the engineering design process.

1.1.2.2. Embodiment design and detail design

The phase of embodiment design (third step, cf. Fig. 2) concerns the development of preliminary technical layouts from the selected concept variants. After an optimization process according to the required specifications, the most promising definitive layouts are selected. The example proposed in Fig. 4 represents the development of the technical layouts starting from the concept variants of Fig. 3. As done in Fig. 3, the purpose of this example is to schematically illustrate the embodiment design, without dwelling on the methods whereby technical layouts are developed, optimized and selected. Regarding the parametric annotation used in Fig. 4, the superscripts “1” and “2” indicate the number of variant and of the corresponding layout, while the subscripts “p” and “d” stand for “preliminary” and “definitive”, respectively.

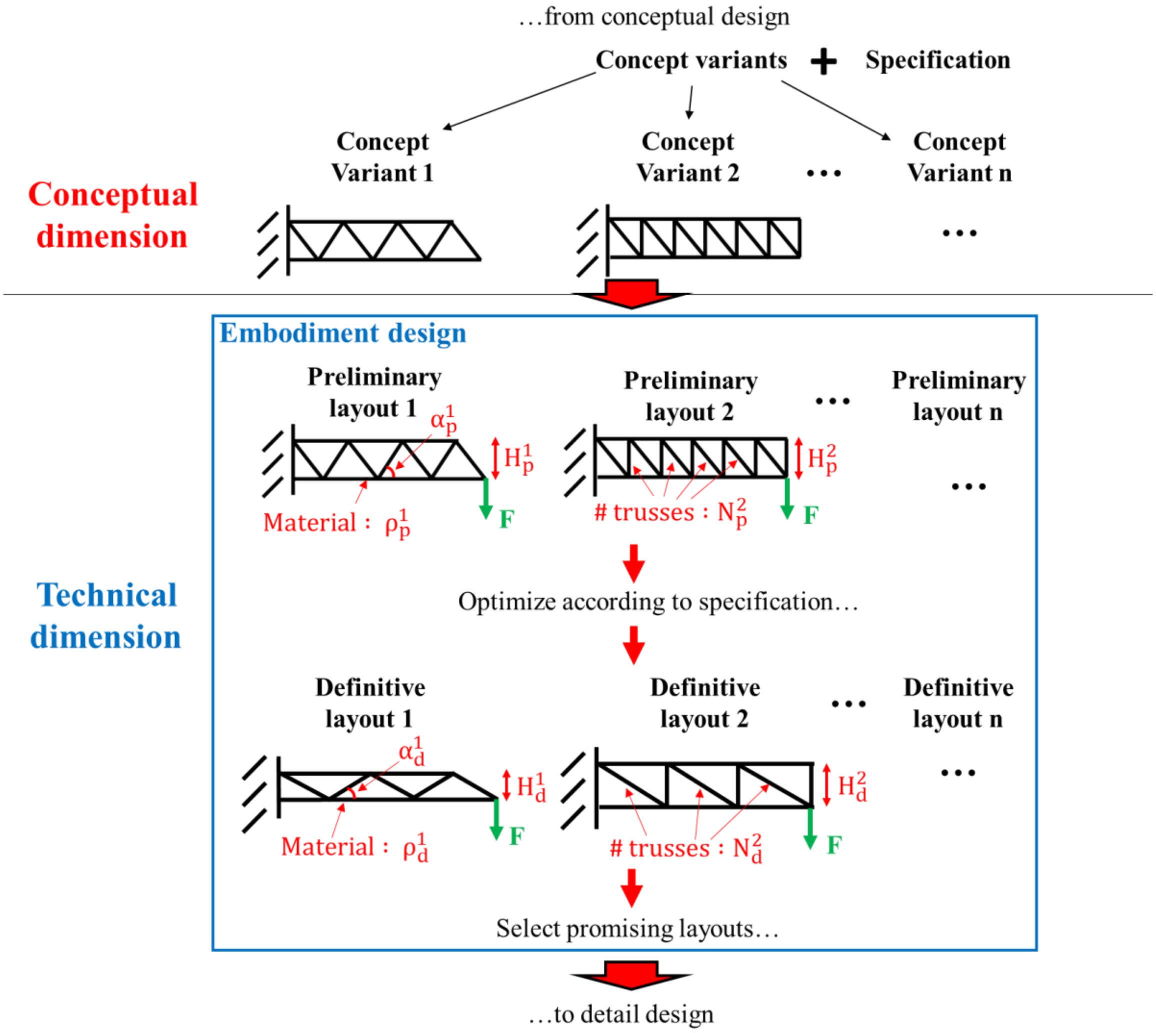


Fig. 4: Schematic example of embodiment design.

As shown in Fig. 4, the development from the conceptual dimension of variant to the technical dimension of layout requires the definition of some preliminary numerical values, regarding for example:

- Geometry, as lengths (H), angles (α) or the number of internal trusses (N).
- Material properties, as the density (ρ)
- Work conditions, as loads (F) and constraints.

In general, this means that the further development of the studied layouts is constrained to the variation of the chosen parameters. Hence, the optimization enables to obtain their definitive (best, optimal) numerical according to the required specifications. As reported in Fig. 1, the optimization process will be further deepened in section 1.3. Finally, the most promising definitive layouts pass to the phase of detail design (fourth step, cf. Fig. 2), which concerns the completion of the design documentation, as the production drawings. According to [16], even if the high-level decisions are normally taken during conceptual and embodiment design, designers should not relax too early, since any lack of attention in the design documentation could ruin even the best concepts.

1.1.2.3. Final considerations

The three design roles reported in Table 1 and the model proposed in Fig. 2 constitute our reference for the design process in this work. Thanks to Fig. 3 and Fig. 4, we can also highlight the fundamental difference between the conceptual dimension, where everything is potentially possible, and the technical dimension, where everything is constrained to the chosen preliminary conditions.

1.2. Purpose

The current section presents the purpose of the thesis and is organized as follows (cf. Fig. 1). The concept of innovation is analysed in section 1.2.1. In order to formulate the BDPs, section 1.2.2 focuses on breakthrough innovations (cf. Fig. 1). Afterwards, section 1.2.3 concerns the cognitive biases. Finally, section 1.2.4 introduces the purpose and the motivation of the thesis.

1.2.1. The concept of innovation

Over the twentieth century, the concept of innovation has become very popular in the industrial and entrepreneurial world, especially when linked to technological development [13]. Notably, the term "innovation" has been used in different contexts over history and it has assumed diverse meanings. Since its Latin origins (*innovare*) until the twelfth century, it denoted a change (*novus*), something new and young. During the sixteenth century, the sense of innovation moved to what is original, unexpected and involves creativity. This aspect is still present nowadays.

1.2.1.1. The contribution of Schumpeter before 1960

Before 1960, scholarly publications on innovation were few and far between [19]. The main exception is represented by the social scientist Schumpeter, in the first half of the twentieth century. Notably, he studied the role of innovation and entrepreneurship in economic growth, with an original approach based on economics, sociology and history insights. In his early works, Schumpeter firstly theorised invention as a process of recombinant research. To him, an invention is a "new combination" (an untried possibility) of existing ideas, processes or technologies [20], and it can be "an idea, a sketch or a model for a new improved device, product, process or system" [21]. Consequently, Schumpeter defined the entrepreneur as a pioneer who carries out new combinations and exploits them to reform the pattern of production and introduce a revolutionary social, economic or technological change, i.e. an innovation [19,20]. According to his definition, innovations can concern new commodities, new methods, new forms or organizations, new sources of supply and new markets [13]. Schumpeter further focused on the economic and technological aspects. He argued that, in the economic sense, an innovation is accomplished "only with the first commercial transaction involving the new product, process, system or device" [21]. On the technological side, he implemented different models to describe the technological change. The following model is the most basic and influential: invention ➔ innovation ➔ diffusion [13,22]. According to his vision, among all the implemented inventions, very few of them lead to innovations. These latter ones, which represent successful new technologies, are thus diffused until their use is widespread [22]. In his later works, Schumpeter extended his approach to also take into account organized R&D (Research and Development) activities in large firms [19,20].

1.2.1.2. Freeman's revolution after 1960: innovation as commercialization

Since the early 1960s, the number of innovation studies has started to grow significantly, due to an increasing interest in Schumpeter's ideas, as well as in technological innovation and R&D activity. Fig. 5, adapted from [19], shows the growth of the social science articles which contain the word "innovation" in the title, from 1955 to 2006. The vertical axis (left side of Fig. 5) reports their ratio (in per cent) to the total number of social science articles, while the legend (right side of Fig. 5) reports the respective authors' disciplinary backgrounds in per cent.

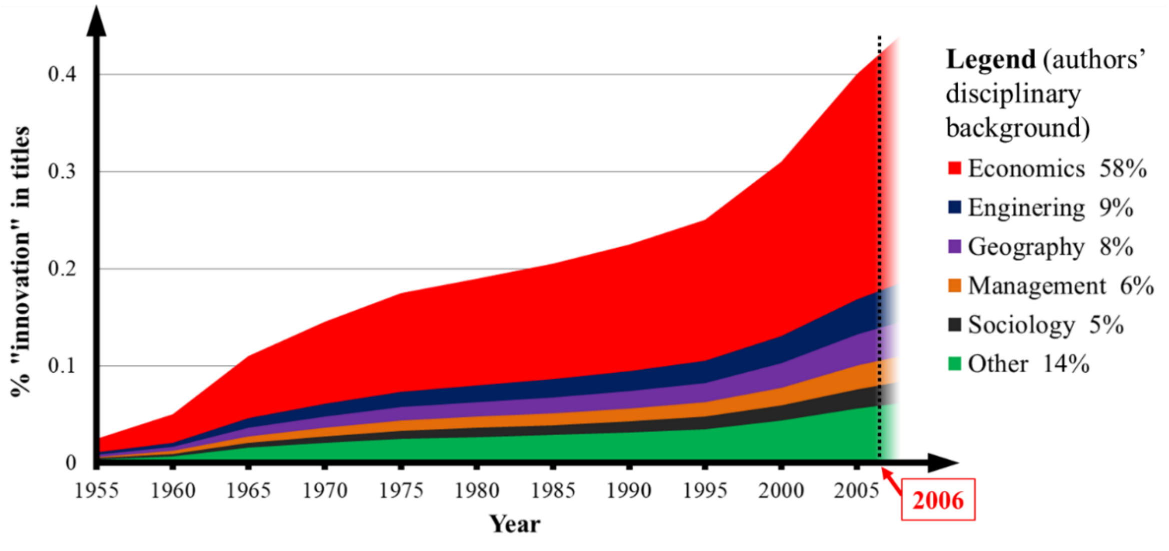


Fig. 5: Growth of the social science articles containing 'innovation' in the title (in per cent) and authors' disciplinary background (cf. [19]).

As illustrated in Fig. 5, the number of these articles is continually increasing since 1960, and a more accentuated slope growth can also be remarked since the late 1990s. Moreover, the legend clearly indicates the predominance of the economic background among the social science authors, followed by engineering and geography. This highly skewed background distribution is probably due to the original contribution of Freeman, who was one of the most influential scientists in the field of innovation [13,19]. The core aspects of Freeman's work are the representation of innovation as commercialization and the definition of technological innovation as product commercialization [13]. A first important break point with the past, and in particular with respect to Schumpeter, is the strong meaning of commercialization which the innovation concept assumes in Freeman's analysis. As anticipated, for Schumpeter innovation firstly consists in combining and introducing a change ("any doing things differently") [13], even if he was aware that an innovation can become such only after commercialization [21]. However, he did not discuss his ideas in terms of commercialization, except for innovations regarding new commodities [13]. A second important break point concerns the difference between the traditional view and Freeman's view of technological innovation. The tradition on technological change had already emerged in the 1930s. Technologies were seen as a source of unemployment by sociologists, as in the previous century, but as a source of productivity by firms and economists. With such a focus on productivity, technological innovation was concerned with the use of technological inventions in industrial production. For instance, Schumpeter formalized his idea of technological innovation in the sense of combining factors of production in a new way [13]. Between the 1960s and the early 1970s [21,23], Freeman initiated a new tradition, moving the focus from productivity to market: to him, the technological innovation is the commercialization of technological inventions, for either customers (as products) or firms (as processes) [13,19]. According to this view, the inventor becomes a businessman who commercializes a new product. On the other hand, the central role and economic importance of the commercialization aspect affect the meaning of innovation, which is usually understood as technological innovation according to this new tradition.

1.2.1.3. Freeman's analysis of R&D: the introduction of the national policy component

The R&D, intended as an activity aimed at technological innovation, is another relevant subject of Freeman's contribution. The introduction of the national policy component in his analysis represents a discontinuity with the past. Till then, technological innovation had been discussed in disciplinary terms, mainly sociological (unemployment) and economic (productivity). Freeman focuses on market and policy failures in relation to the technological innovation of consumer goods and services. Notably, he remarks how the vast majority of public R&D is devoted to national security and prestige R&D (military, nuclear, space), especially in his epoch, characterised by the Cold War. Moreover, he highlights how "the present R&D selection project techniques are biased overwhelmingly towards technical and short-term competitive economic criteria" [21]. To Freeman, the reallocation of R&D resources must be the main concern of national policies, which should support science and technology based on their contribution to social welfare. To this purpose, he proposes several strategies to involve criteria of user-oriented innovations (satisfaction, environment, aesthetics, social benefits) in the R&D processes: standards, regulations, public representation on committees and technology assessments [13,21].

1.2.1.4. The intergovernmental standpoint of the OECD and its principal international reports

Since the early 1960s, intergovernmental organizations have been the principal diffuser of Freeman's ideas. A discussion on national coordination of science had already started in England in the first century half, and then spread all over Europe in the 1950s. Since its foundation in 1961, the OECD (Organization for Economic Cooperation and Development) encourages international policies to establish the role of science and technology in economic development. The early OECD reports also focus on lags and gaps in science and technology between European countries and the US (United States), which became a model of economic growth, technology development, productivity and market share [13]. The Frascati manual [24], one of the most important OECD documents, constitutes a world recognised standard for R&D study and it is widely used by several organizations associated to the United Nations and the EU (European Union). It contains some fundamental definitions (as of base research, applied research, experimental development) and it concerns the measurement of R&D resources (costs and staff). The first edition (1963) was based on a document presented by Freeman, who acted as consultant to the OECD [13]. The OECD is also responsible for a deeply developed discussion on innovation that still continues today. Since the early reports in 1966 [25] and 1970 [26], the OECD retained a vision of technological innovation as commercialized invention, and further implemented this representation in the following decades. The Oslo manual [27], first released in 1992, defines and distinguishes the technological innovation of product and process. According to the Oslo manual, the product innovation concerns the commercialization of a more performing product that provides objectively new or better services to consumers. On the other hand, the process innovation concerns the adoption of new or considerably better production or distribution methods. The technological innovation of process can involve changes in materials, human resources or work methods, contemporarily or separately.

1.2.1.5. The crystallisation of innovation as product commercialization

Between the 1960s and 1970s, other public reports sharing the same OECD innovation perspective were released by other important public organizations, as the US Department of Commerce and the UK Central Advisory on Science and Technology. Despite the development of other definitions in sociology, management and political science, only the definition of commercialization has been selected and standardised by OECD and governments. The reason lies in the relevance of commercialization to policy purposes (market shares). However, this contributed to a crystallization of the concept of innovation as technological innovation and on a focus on firms and economic growth as ultimate outcome [13]. Few studies discuss what innovation is. On the contrary, the majority retains the specific perspective of technological innovation and specializes on firms' organizational features and market to optimize product commercialization. Innovation's social and institutional aspects are studied for their contribution in the innovative performance of firms, while the social issues, as those criticized by Freeman (cf. section 1.2.1.3), occupy a marginal position [13].

1.2.1.6. Final considerations: our perspective of innovation in relation to design

We have investigated the evolution of the innovation concept from the 1960s to the recent decades, from an industrial and entrepreneurial standpoint. As argued, a different interpretation has been added to this concept according to the historical period. We have seen how innovation has nowadays acquired a strong economic and technological meaning, mainly due to the international reports of the OECD. We believe this is one of the main reasons why this concept has become so popular even in the design field. The principal meaning that innovation has assumed according to the different epochs are reported in Fig. 6, with the support of a timeline (left side). The relationship with the concept of "new" has been also highlighted (cf. Fig. 6, right side).

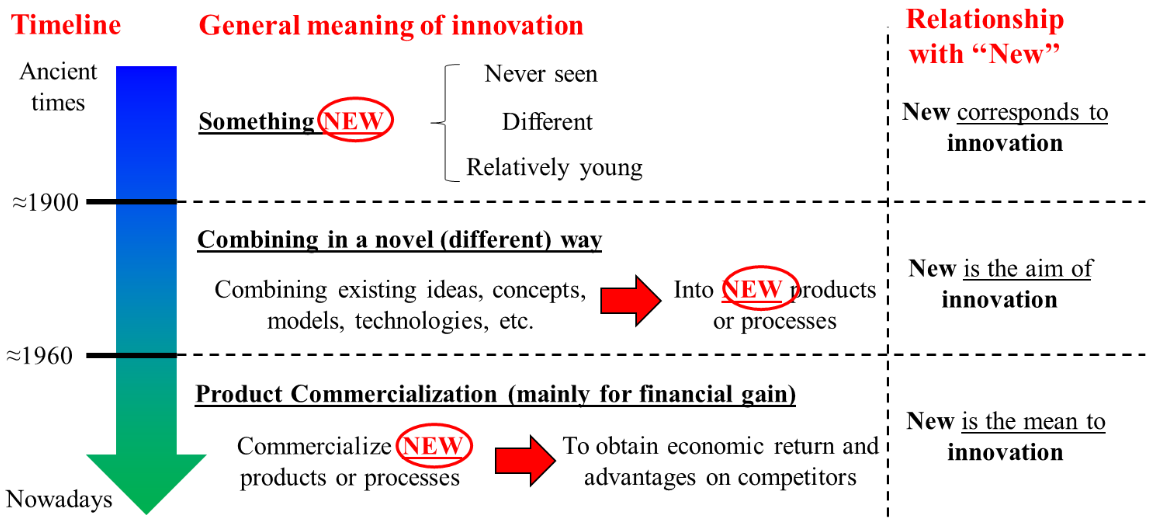

Timeline
General meaning of innovation
Relationship with "New"
Ancient times
Something NEW
Never seen
Different
Relatively young
New corresponds to innovation
≈1900
Combining in a novel (different) way
Combining existing ideas, concepts, models, technologies, etc.
Into NEW products or processes
New is the aim of innovation
≈1960
Product Commercialization (mainly for financial gain)
Commercialize NEW products or processes
To obtain economic return and advantages on competitors
New is the mean to innovation
Nowadays

Fig. 6: Principal meaning assumed by innovation according to time and relationship with "new".

Even if the economic interpretation is the most predominant vision, all the other meanings (cf. Fig. 6) still exist and are currently used in different contexts by innovation scholars [10,13,19]. This could explain the great vagueness surrounding this concept. For the purpose of this work, we adopt a general definition of innovation based on Schumpeter's vision: a novel combination with the power to produce a change (cf. Fig. 6 and section 1.2.1.1). This definition is reported in Fig. 7, where three key aspects are highlighted and explained in relation to our design focus.

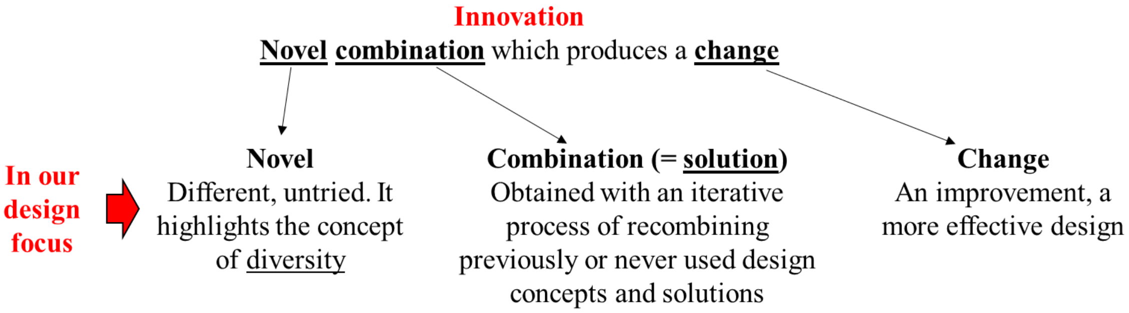

Innovation
Novel combination which produces a change
In our design focus
Novel
Different, untried. It highlights the concept of diversity
Combination (= solution)
Obtained with an iterative process of recombining previously or never used design concepts and solutions
Change
An improvement, a more effective design

Fig. 7: Key aspects of the adopted definition of innovation in relation to our design focus.

Notably, the diversity of the generated combinations, as underlined in the term "novel" (cf. Fig. 7), represents for us a very important point of the given definition. We believe a design approach should insist on this aspect, by helping designers admitting different standpoints and by proposing different solution options ("combination" cf. Fig. 7). These ideas are embodied in the BDPs, which will be introduced in section 1.2.2. The term "combination" (cf. Fig. 7) identifies a generic design solution and highlights the recombinant and iterative nature of the design process (cf. Fig. 2). Finally, as reported Fig. 7, we intend the produced "change" in a positive sense, as an improvement of a previous design condition.

1.2.2. The "Breakthrough" Design Practices (BDPs)

According to the adopted definition (cf. Fig. 7), an innovation is considered such if it produces a (positive) change. This change can be more or less significant, predictable or unexpected. Many efforts have been aimed to classify innovations by describing and "measuring" the produced change [2].

1.2.2.1. A qualitative model to classify innovations

Over the last decades, scholars have employed different adjectives, often imposing a bimodality to innovation: continuous versus discontinuous, evolutionary versus revolutionary, and incremental versus (breakthrough). As argued in [2], these adjectives represent an indicative classification method. Different models have also been implemented to classify innovations in a more structured way, mostly from an economic point of view. The model proposed in Table 2 (cf. [28]) is one of the most used, and highlights two principal dimensions of industrial innovation: newness of technology and customer need fulfilment. The first one represents a measure of how much a technology involved in a new product is different from prior technologies. The second one represents a measure of the benefit given to customers by a new product and its capability to create a new market.

Table 2: Types of product innovations (cf. [28]).

		Customer Need Fulfilment	
		Low	*High*
Newness of Technology	*Low*	Incremental Innovation	Market Breakthrough
	High	Technological Breakthrough	Radical Innovation

According to Table 2, an incremental innovation entails relatively moderate changes in technology and customer benefit. Market breakthroughs are based on core technology, similar to existing products, but provide higher costumer benefits. Vice versa, a technological breakthrough adopts a substantially different technology but does not provide superior customer benefits. Finally, radical innovations involve substantially new technologies and provide substantially higher customer benefits [28]. The qualitative nature of this model (*low* and *high*, cf. Table 2) suggests that there is not a well-defined border between incremental and breakthrough (or radical) innovations and, as argued in [2], that innovation can be thought as a continuous space [2].

1.2.2.2. Concept originality: the key to breakthrough innovations

The use of new technologies and the birth of new markets (cf. Table 2) are of course fundamental variables to consider, but they do not constitute the unique term of classification. As observed in [2], a large variety of innovations employed existing technologies and concerned already mature markets, as the Sony Walkman, for example, which still resulted in an economic boom for the company. This product had a great success and a large diffusion, especially among young people. In this case, the concept of the product, i.e. the possibility to bring along your own music (cf. section 1.1.2.1), represented the real breakthrough innovation. Another similar example is reported in [17]: in 2001, the BMW Group decided to conceive a new interface device for controlling a manifold of functions in luxury cars. This interface device was based on an already mature technology, the joystick, very important for the video game industry, but still resulted in a great success. Notably, the breakthrough was determined by transfer of the concept of joystick (cf. section 1.1.2.1) from the video game field to the automotive domain. Based on these examples, we can make the following important considerations:

- As also argued in [2,29], the breakthrough potential of an innovation is embodied in its concept. Notably, the originality of the concept represents the key to breakthrough innovations.
- Consequently, the phase of conceptual design (cf. Fig. 2), in which concepts are born, is of crucial importance for the achievement of breakthrough innovations.

1.2.2.3. Incremental and breakthrough innovation in a design process

We can better outline the concepts of incremental and breakthrough innovation in our design context by means of Fig. 8. Remembering that design is an iterative process of combination (cf. Fig. 2 and Fig. 7), the histogram in Fig. 8, adapted from [1], shows the typical relation between the number of carried combinations (vertical axis) and their design outcome (horizontal axis). The latter represents a statistical measure based on patent citations, scientific citations, financial returns and number of times a novel combination is used by future designers. As argued in [1], every well-sampled of design outcome based on these parameters shows an highly skewed profile as in Fig. 8.

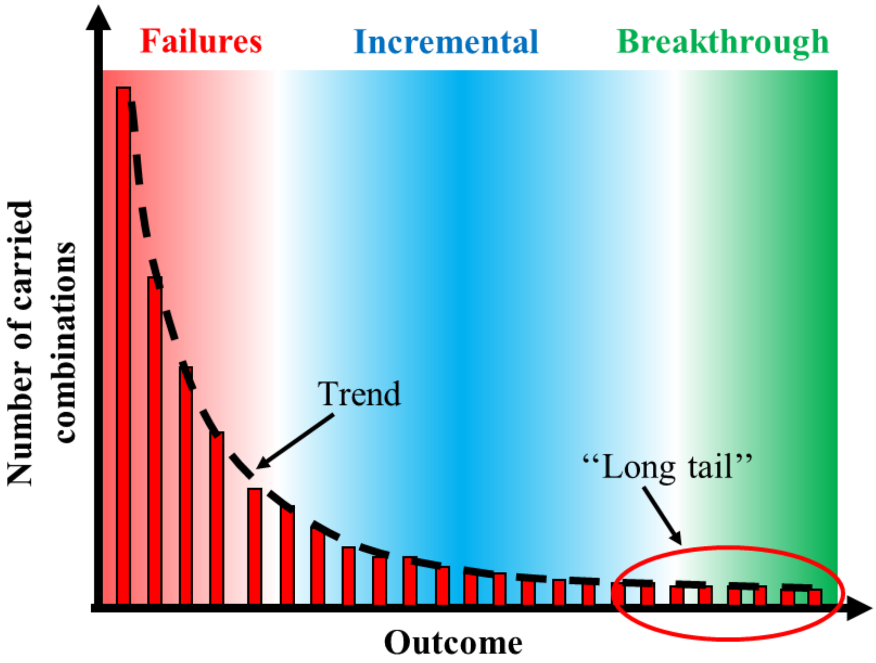


Fig. 8: Number of carried combinations vs design outcome (cf. [1]).

The histogram suggests that the most part of the carried combinations result in a very poor outcome, as indicated by the great density on the left. For these combinations, the outcome is so low that they can be considered as failures (cf. Fig. 8). Few combinations have a moderate value (incremental cf. Fig. 8), while the very outliers on the right represent the true breakthroughs and form the "long tail" of innovation [1]. Based on these considerations, a design process could produce a great amount of useless or minor new combinations before achieving a breakthrough innovation. The continuous trend in Fig. 8 indicates that the innovation categories (failures, incremental and breakthrough innovations). cannot neatly distinguished. This means there are no defined patterns which guarantee to achieve breakthrough innovations. However, their birth can be favoured by observing some principles during the design process. We called these principles "Breakthrough" Design Practices (BDPs).

1.2.2.4. The base of the BDPs

The BDPs are directly related to the nature of the human subconscious tacit knowledge [2,28]. The latter is a highly personal and unique form of knowledge, and it derives from experience, practice, perception and learning. Based on these characteristics, a designer can contribute to innovation in different ways [29]. In the model of tacit knowledge proposed in Table 3, adapted from [2], the tacit skills correlated to the innovative abilities have been distinguished into technical and cognitive skills. As shown in Table 3, the experience gained through "learning-by-doing" gives the lowest innovative abilities. This type of knowledge, for example, allows an expert workman to be more productive than a novice, despite the same tools and a similar training. A combination of formal education and work experience (average level, cf. Table 3) awards more significant innovative abilities. This category can be represented by a specialist engineer, endowed with problem-solving skills but limited by his preference for details [2].

Table 3: How the tacit knowledge contribute to innovation (cf. [2]).

Innovative abilities		**Correlated tacit skills**	
Level	**Description**	**Technical**	**Cognitive**
Low	Tangible results achieved through bodily actions	Learning-by-doing experience	Pattern recognition
Average	Creative solutions to specific problems Innovations in techniques, methods and processes	Educational and experienced specialization	Problem-solving Mental models
High	Breakthrough conceptual solutions (cf. section 1.2.2.2)	Deep and broad multidisciplinary specialization	System thinking Intuition Insight

The highest level of innovative ability indicates the skills which more contribute to breakthrough conceptual solutions (cf. section 1.2.2.2):

- On the technical side (cf. Table 3), a deep specialization in more disciplines allows to analyse problems under different points of views, and thus to broaden the search range in the solution space.
- On the cognitive side, breakthrough innovators have an overall view of the solution space (system thinking, cf. Table 3): they can see and originally exploit (intuition and insight, cf. Table 3) the interrelationships between different technologies, or even different and apparently unrelated disciplines, transcending the technical details and visualising the final solution.

These skills are the base of the BDPs and represent the essence of creativity [2,29]. In order to formulate the BDPs, we want to show how these technical and cognitive skills, respectively, can be released and exploited.

1.2.2.5. The BDPs based on the technical skills

We can better illustrate this aspect by assuming the point of view of a manager, who organizes and supervises the design process. Notably, managers are charged to monitor and measure the quality of the design process. Based on the breakthrough technical skills reported in Fig. 9, managers can promote a multidisciplinary collaboration between more designers with different fields of expertise [1,30]. Fig. 9 shows the effect of promoting multidisciplinary collaboration on the design outcome, according to the typical skewed distribution in Fig. 8. For brevity, the category of incremental innovations (blue background, cf. Fig. 8) has been removed.

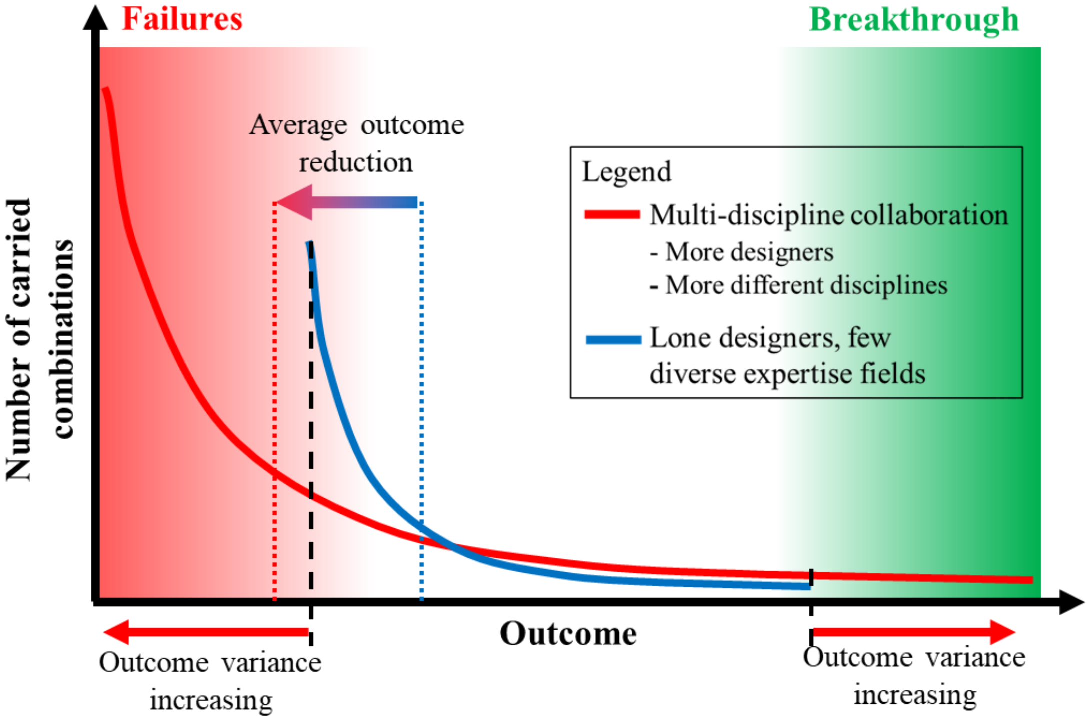


Fig. 9: Effect of promoting multidisciplinary collaboration (cf. [1,30]).

As shown in Fig. 9, a greater diversity between the specialization fields of collaborators tends to reduce the average outcome of the design process and, for this reason, it can be very risky. However, the more this diversity is significant, the more the variance of the inventive outcome increases. This means that much worse combinations (failures) but also much better ones (breakthroughs) are more probable. The behaviour reported in Fig. 9 allows to formulate the two following BDPs:

BDP a. Consider multiple and different solution paths. Based on Fig. 9, the probability to achieve more interesting (breakthrough) solutions increases by carrying out more and different combinations. Given a design problem with some design requirements to satisfy, we can admit mechanical, physical or chemical solutions (i.e. from different disciplines) [1,30]. We can also consider multiple and different solutions within the same discipline [2], as shown in the example of Fig. 10. The solutions listed in Fig. 10b allow to reduce the vertical displacement of the cantilever beam in Fig. 10a.

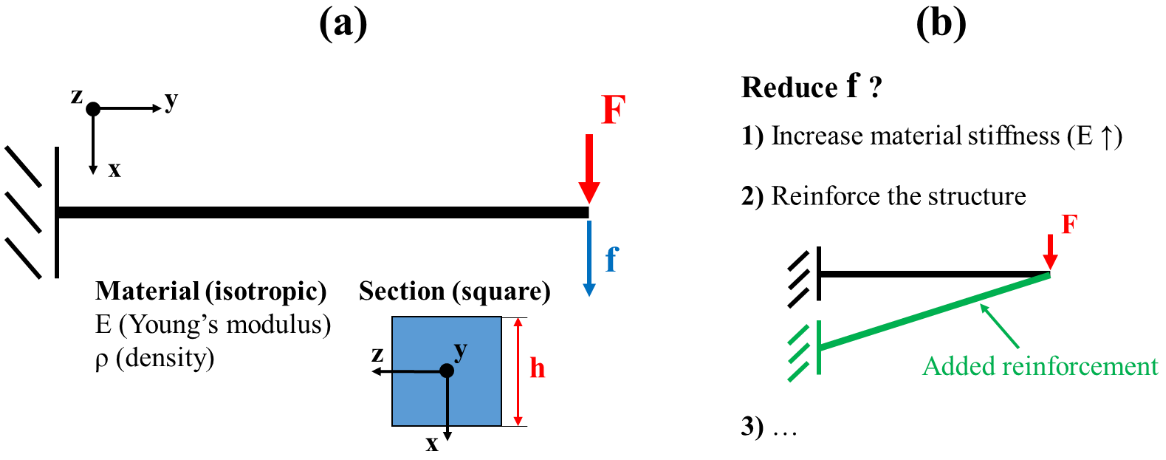


Fig. 10: Example: (a) cantilever beam and (b) different mechanical solutions to reduce f.

All the solutions in Fig. 10b are mechanical (same discipline), but they act on the structure in a different way. Indeed, solution 1 suggests increasing the material stiffness, while solution 2 involves the addition of a reinforcing element which modifies the configuration of the original system (cf. Fig. 10b).

BDP b. Consider good as well as bad solutions (keep also bad solutions). Bad solutions are those which poorly improve the outcome (failure, cf. Fig. 9). According to [1,3,30], they should not be discarded, since they could be combined with other solutions in the future and result in breakthroughs.

1.2.2.6. The BDP based on the cognitive skills

Keeping the managerial perspective, managers' role cannot be limited to assemble and organize the team. As argued in [29], the breakthrough cognitive skills (cf. Table 3) need targeted stimuli to be released and expressed in a tangible way: designers' creativity must be awakened. This represents one of the hardest challenges for managers. Of course, we cannot exclude those stimuli could also randomly and unexpectedly occur. Nevertheless, managers can observe the following principles in order to achieve this objective:

- Encouraging designers to "think outside the box", which means pushing them out of their comfort zone by inspiring them to explore new and unconventional paths. Designers should be enabled to "wander with their minds" without imposing pre-defined structures and to carry out "wild" ("foolish") new combinations (which increase the outcome variability, cf. Fig. 9) by aiming to ambitious targets [1,30,31]. Fostering a personal and emotional engagement of designers in the design process could facilitate this task [2].
- Create a stimulating environment: aligning all minds to the same focus and encouraging an atmosphere of mutual feedback and learning. The team should operate as a unified entity, ideas should constantly flow from everyone, including the manager, to each other. According to [2], this task could be facilitated by promoting direct and continuous personal interactions with frequent meetings, and by avoiding the indirect communication forms, as e-mails and documents, as much as possible.

These points are summarised in Table 4.

Table 4: Managerial principles to release the breakthrough cognitive skills.

Principle	Description	Personal aspect
Push designers out of their comfort zone	• Encourage creative improvisation: trying unconventional solution paths without imposing pre-defined structures • Fix ambitious targets	Favour the emotional engagement in the design process
Create a stimulating work environment	• Align all designers to the same focus • Encourage mutual feedback and mutual learning	Favour direct communication forms, increase personal interactions

Based on these principles, we can thus formulate the following BDP:

BDP c. Creative improvisation. Mostly during conceptual design (cf. Fig. 2), the designer should be free to "improvise", i.e. trying unconventional and instinctive solution paths without following pre-defined patterns and design methodologies (pre-defined structures, cf. Table 4), as for example those proposed in the informatic CAD software. As argued in [32,33], these tools force designer to focus on technical details linked to their functioning, "forgetting" their real target. On the other hand, improvising enables designer to transcend technical details and to have an expanded overview of the solution space (system thinking, cf. Table 3). This also improves his ability in exploiting eventual interrelationships between different combinations (insight, intuition, cf. Table 3) to achieve breakthrough innovations.

1.2.2.7. Final considerations

The three BDPs formulated in sections 1.2.2.5 and 1.2.2.6 are reported in Table 5, where their fundamental concepts are also highlighted.

Table 5: Breakthrough design practices (BDP).

BDP	Fundamental concepts
a. Consider multiple and different solution paths	The probability to achieve more interesting (breakthrough) solutions increases by carrying out more (multiplicity) and different (diversity) combinations (cf. Fig. 9)
b. Consider good as well as bad solutions	Keep also bad solutions (do not discard them a priori). They could be potentially recombined with other solutions and generate a breakthrough
c. Creative improvisation	Try unconventional solution paths without following pre-defined patterns and methodologies, transcend technical details

As anticipated in section 1.2.2.3, these BDPs cannot guarantee the achievement of breakthrough design solutions. However, if observed, they enable designer to enormously enlarge the solution space, increasing the possibility to include breakthrough solutions. Unfortunately, the indications provided by the BDPs are usually unknown or ignored [2]. Instead of BDPs, designers often tend to follow “bad design habitudes”, due to some mental mechanisms strongly rooted in the human mind: the cognitive biases [3]. We will focus on them in in section 1.2.3.

1.2.3. Cognitive biases: the greatest obstacle against breakthrough innovation

The cognitive biases are subconscious mental mechanisms which continually influence our thoughts and decisions, and prevent us to think with a “completely open mind” [4,5]. We want to study them in order to understand how they impact on our design abilities.

1.2.3.1. The effects of cognitive biases on our behaviour

By nature, our brain tries to keep us safe and to preserve energy. In a sense, this characteristic makes us “lazy”: it subconsciously creates patterns of thinking and behaviours which lead us to assume as “dangerous” all that is new and unknown [4,5]. As argued in [4], this aspect allowed us to survive several times during our evolution, but it represents one of the greatest obstacles to the application of the BDP. Notably, when we are faced to a problem, our brain has already determined which is the best chance of success even before we start to reflect. Indeed, the brain operates on autopilot and is used to automatically taking well-known shortcuts, rather than consuming energy to explore new paradigms. As reported in [34], “we think of ourselves as (being) in the driver’s seat, with ultimate control of the decisions we make – but, alas, this perception has more to do with our desires than with reality”. A description of the principal cognitive biases is reported in Table 6 according to [4,5,7], in the form of relationship between their cause and their effect on our behaviour. Based on Table 6, not only cognitive biases are innately eradicated in the human being, but they are also determined by the received education. Notably, the education contributes to form familiar patterns (functional fixedness, cf. Table 6) and to outline one’s preconceptions (confirmation, cf. Table 6) [3,7].

Table 6: Causes and effects of the principal cognitive biases.

Cognitive bias	Cause	Effect on behaviour
Status quo	The brain hates anxiety and uncertainty, and tends to maintain an equilibrium state to save energy	We tend to reject a priori what is uncertain, because too challenging, too risky and it requires too much time
Functional fixedness	The brain loves the familiarity of the patterns on which we have been trained and educated	We are limited to use objects in a traditional way, it is hard to think "outside the box"
Confirmation	We tend to interpret the reality, make assumptions, and remember information seeking confirmation and support to our preconceptions	We tend to dismiss, refuse and ridicule the alternatives which do not fit with our view, also because we simply dislike them

1.2.3.2. Biased design habitudes (BDHs): how cognitive biases affect design thinking

The influence of cognitive biases (cf. Table 6) on design thinking represents an important issue regarding the achievement of breakthrough innovations. According to [3], cognitive biases are responsible for biased design habitudes, denoted BDHs. Based on [3,5,6,32,33,35] and on Table 6, Table 7 shows the contraposition between the BDPs (i.e. what designers should do to achieve breakthrough innovation, cf. Table 5) and the most common BDHs (i.e. what designers instead tend to do), highlighting also the responsible cognitive biases. Due to BDHs, the designer is led to prematurely terminate the design process after evaluating a poor set of solutions, impairing the generation of new ideas and the testing of different hypotheses. Many aspects of the BHDs derive from the design education. As argued in [6,32,33,35], the traditional engineering pedagogy teaches science theory focusing on the technical aspects. Notably, design students are trained to eliminate ambiguity, minimizing all uncertainties by defining fixed parameters, to quickly find optimal compromise solutions (functional fixedness, cf. Table 7). Moreover, they are not used to work in team, and they are thus reluctant in considering different points of view which do not fit with their preferences (confirmation, cf. Table 7). As visible in Table 7, all cognitive biases simultaneously contribute to prevent the BDP a, while the BDP b and c are essentially obstructed by functional fixedness.

Due to BDHs, the designer is led to prematurely terminate the design process after evaluating a poor set of solutions, impairing the generation of new ideas and the testing of different hypotheses. Many aspects of the BHDs derive from the design education. As argued in [6,32,33,35], the traditional engineering pedagogy teaches science theory focusing on the technical aspects. Notably, design students are trained to eliminate ambiguity, minimizing all uncertainties by defining fixed parameters, to quickly find optimal compromise solutions (functional fixedness, cf. Table 7). Moreover, they are not used to work in team, and they are thus reluctant in considering different points of view which do not fit with their preferences (confirmation, cf. Table 7). As visible in Table 7, all cognitive biases simultaneously contribute to prevent the BDP a, while the BDP b and c are essentially obstructed by functional fixedness.

Table 7: Contraposition between BDP and BDH, and corresponding cognitive biases.

BDP What designers should do	BDH What designers instead do	Responsible cognitive bias
a. Consider multiple and different solution paths	Choosing what can be quickly imagined and realized	Status quo: we tend to reject seems too challenging and requires too much time Functional fixedness: designers are trained to quickly find optimal compromise solutions
	Dismissing unliked solution paths Ignoring disconfirming data	Confirmation: we tend to refuse what does not fit with our preconceptions (or what we simply dislike) Status quo: disconfirming data perturbate our equilibrium and cause anxiety
b. Consider good as well as bad solutions	Excessive focus on good (optimal) solutions	Functional fixedness: designers are trained to discard bad solutions
c. Creative improvisation	Excessive focus on technical aspects	Functional fixedness: designers are very familiar with technical aspects; they are not trained to “think outside the box”

Based on Table 7 and Table 6, the main effect of cognitive biases on our design thinking is schematically illustrated in Fig. 11, where the big circle is the space of the solutions we could potentially imagine, while each round tick represents a different solution.

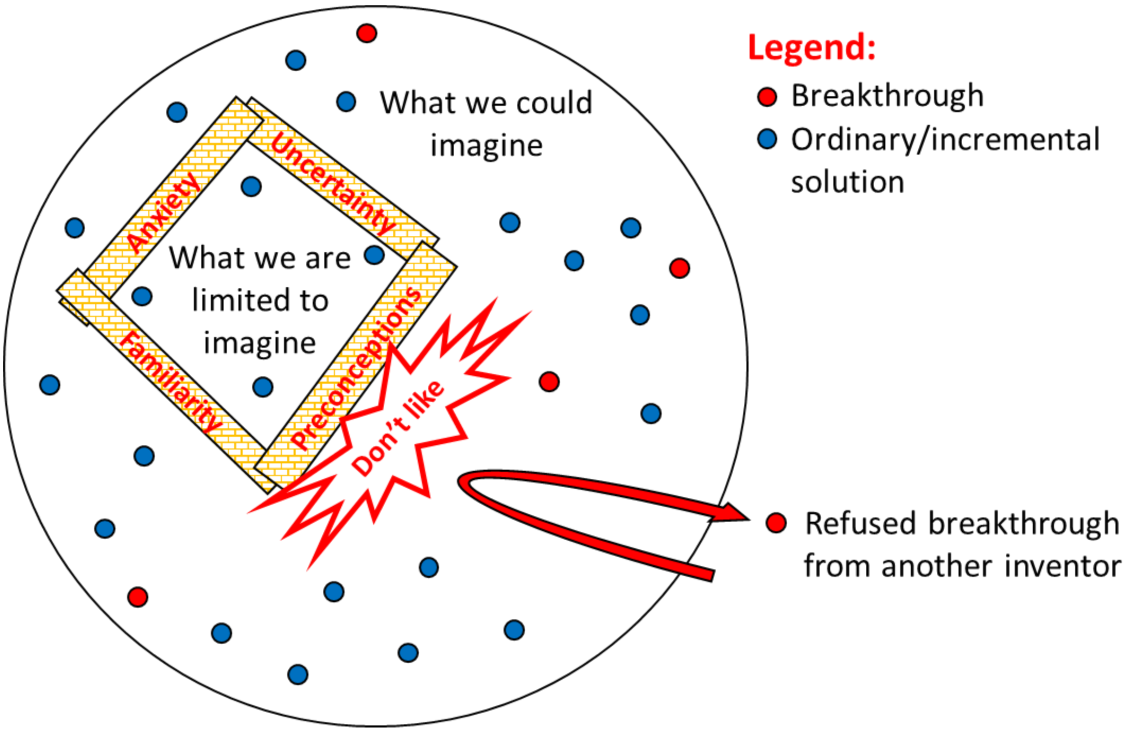

What we could imagine
What we are limited to imagine
Anxiety
Uncertainty
Familiarity
Preconceptions
Don't like
Legend:
● Breakthrough
● Ordinary/incremental solution
● Refused breakthrough from another inventor

Fig. 11: Restriction of the solution space due to cognitive biases.

The cognitive biases, by means of the BDHs, preclude us the possibility to imagine different and more interesting things (breakthroughs, cf. Fig. 11), Contrary to the BDPs (cf. Table 5), this results in restricting the solution space. Over the recent decades, many efforts have been aimed at creating a new design thinking, to mitigate the cognitive biases (and so the BDHs). The analysed literature contributions focus on two main points of view: management and education.

1.2.3.3. The mitigation of cognitive biases from the managerial perspective

According to [2–5,7], managers can mitigate the confirmation bias (cf. Table 6) by promoting the open discussion (cf. Table 4) of different points of view, to limit egocentrism phenomena. To this purpose, several methods, as brain-storming and 5W1H[1] (cf. [36]), can be very useful to foster socialisation, but may not give a significant impulse to breakthrough innovation. As suggested in [2,3], managers should adopt fast prototyping methods:

- Fast prototyping. Solution samples are immediately constructed in the early steps of the design process (cf. Fig. 2), in which technical requirements are not fully defined yet. This "forces" designers improvising to immediately imagine a physical solution proposal without focusing on the technical aspects, and thus mitigating the functional fixedness (creative improvisation, cf. Table 7). Thanks to fast prototyping, team designers can also openly discuss on visible and tangible prototypes, and this helps mitigating the confirmation bias. Moreover, for commercial design, fast-prototyping easily favours the interaction with costumers, and this offers the possibility to learn from failures and to collect complete data sets [6]. For example, at Sony, the early technical samples, which will serve as "physical proposals", are immediately constructed in the first design phases (cf. Fig. 2). Starting from the approved samples, project teams carry on by creating commercial prototypes, each one conceived to facilitate the communication with consumers and among the same design teams at a specific phase of the design process [37].

The analysed aspects are summarised in Table 8.

1.2.3.4. The mitigation of cognitive biases from the educational perspective

The Product Design Engineering (PDE) program (cf. [32]), at Swinburne University of Technology (Melbourne, Australia), and the Mechanical Engineering 310 (ME310) course (cf. [33]), at Stanford University (California, USA), represent two important educational examples of forging creative designers with a novel approach oriented towards breakthrough innovation. The effects of the traditional engineering pedagogy on the confirmation and functional fixedness biases are reported in Table 7 (cf. section 1.2.3.2). Contrary to the traditional programs, these courses are based on a "learning in action" approach and consist in designing and producing a prototype from a given need of a client. Students are divided into project teams and are expected to search for original solutions, merging their creative efforts in combining new materials and technologies. Due to the ill-defined ("wicked") assignments, all teams

[1] 5W1H: this method intends to analyse problems systematically, according to the essence of the object (What), the essence of the subject (Who, i.e. the manipulator), the problem-existence ways in time and space (When, Where), the reason/principle (Why) and the solution of the problem (How) [36]. In a more improved version, denoted with 5W2H, the term "How" is divided into "How to" and "How much", which respectively indicate the way and the cost to resolve the problem [36].

struggle in defining and re-defining the appropriate engineering requirements and continually re-scope the solution space. Despite the initial anxiety, this enables students to become more flexible and to acquire more advanced skills in handling risks, thus mitigating the status quo bias (cf. Table 6 and Table 7) [32,33]. A particular emphasis is given to hand-sketching and fast prototyping:

- Hand-sketching. In many courses, design is traditionally taught by means of CAD software. Of course, these represent a powerful ally in the advanced design phases. However, in the first phases of conceptual design (cf. Fig. 2), they risk stifling creativity by imposing their methodology to users. Instead, hand-sketching enables students to quickly shift to new alternatives within the solution space, without imposing any technical rule. Therefore, hand-sketching helps to reduce the focus on technical aspects, providing more space for creative improvisation (cf. Table 7, functional fixedness) [38]. Moreover, hand-sketching provides better possibilities of open discussion: each team member can easily participate and "contaminate" the sketch with his ideas. This helps to mitigate the confirmation bias (cf. Table 6 and Table 7) [32,33,39].
- Fast prototyping (cf. section 1.2.3.3). Analogous to hand-sketching, it allows to mitigate functional fixedness and confirmation biases by fostering improvisation and open discussion. By means of fast prototyping, project teams can also frequently interact with experts and clients to iteratively evaluate the engineering and commercial features of their prototypes.

At the end of these courses, the skills acquired by students allow them not only to excel in the resolution of ill-defined problems, but also to overcome other "more technical" students in more technical problems [32,33]. All the analysed aspects are summarised in Table 8.

1.2.3.5. Final considerations

Based on the analysed perspectives, the fundamental principles to mitigate cognitive biases are summarised in Table 8.

Table 8: Fundamental principles to mitigate cognitive biases.

Perspective	Principle	Mitigated cognitive bias
Management	Encouraging open discussion (socialization): • Brainstorming • 5W1H[1]	Confirmation: open discussion limits egocentrism phenomena; different standpoints can be more easily considered
	Fast prototyping ➢ Improvisation to realise early proposals ➢ Open discussion on tangible prototypes	Functional fixedness: more improvisation and less focus on technical aspects (cf. Table 7) Confirmation (see above)
Education	Assigning ill-defined problems	Status quo: students continually "struggle" to achieve the solution, they become more confident in handling anxiety and risks
	Hand-sketching (instead of CAD software) ➢ Less technical rules ➢ Open "contamination" of the sketch Fast prototyping (see above)	Functional fixedness (see above) Confirmation (see above)

Both perspectives highlight the importance of the fast prototyping to foster the creative improvisation and to encourage the open discussion among all the actors of the design process (designers, managers, costumers). We also want to highlight the significant role of hand-sketching, which enables designers to represent and combine even unconventional and incomplete solutions, without following any technical rule.

1.2.4. The purpose of our work

According to Fig. 1, the three fundamental principles reported in Table 9 express the purposes of our work and represent the base of the aided design procedure implemented in this thesis. Table 9 also highlights two important targets involved in all the proposed purposes (common targets).

Table 9: Base principles of the implemented demonstrator, corresponding purposes and common targets.

Base principles	**Corresponding purpose**	**Common targets**
Focus on conceptual design	Favour the birth of original concepts and concept variants (cf. Fig. 3)	Minor focus on optimal solutions Minor focus on technical aspects
Follow the indications provided by the BDPs	Enlarge the solution space as much as possible (cf. Table 5)	
Help to overcome cognitive biases	Prevent cognitive biases to affect the designer's choices and to restrict the solution space (cf. Table 7 and Fig. 11)	

According to our purposes, we want to further comment the chosen principles in view of their common targets (cf. Table 9):

- Focus on conceptual design: The conceptual design constitutes the scope of our procedure. Indeed, as argued in section 1.2.2.2, the concept originality is the key to breakthrough [2,17,29]. As said in section 1.1.2.3, "everything is possible" in the conceptual dimension: the designer is theoretically free from any rule and there is no limit to the creativity of the generated concepts [11,16]. Notably, conceptual design involves a minor focus on technical aspects and on optimal solutions (cf. Table 9), which should be normally considered in the phase of embodiment design (cf. Fig. 2 and Fig. 4).
- Follow the indications provided by the BDPs: We want to offer designer the possibility to enlarge the solution space as much as possible, considering multiple and different solutions, good and bad ones (minor focus on optimal solutions, cf. Table 9), classical and unconventional ones (cf. Table 5). Notably, unconventional solutions can be achieved through the creative improvisation (cf. section 1.2.2.6), i.e. by transcending technical details (minor focus on technical aspects, cf. Table 9) without following pre-defined design methodologies.
- Help to overcome cognitive biases: We want to prevent cognitive biases to affect designer's choices and to restrict the solution space. Based on Table 8, we will insist on some design techniques which allow to mitigate the BDHs (cf. Table 7). Notably, the employed techniques will enable designer to divert the attention from technical design aspects and optimal solutions (cf. Table 9), improving the diversity and the multiplicity of the carried design combinations.

1.3. Approach

According to the BDHs introduced in section 1.2.3.2, designers tend to excessively focus on technical aspects, searching for optimal solutions (functional fixedness, cf. Table 7), i.e. following an optimization approach. As suggested by the common targets in Table 9, our purposes go in the opposite direction, in order to limit these biased tendencies. Moreover, the optimization is generally involved in the embodiment design phase (cf. Fig. 2 and Fig. 4), which is out of the scope of our procedure. However, optimization represents the most widespread approach used by designers [8]. Notably, we believe it is very important to further clarify the reasons why optimization is not suitable to our targets. This will also help to better understand our purposes, before presenting the approach followed in our work. According to Fig. 1, the optimization approach is described and analysed in section 1.3.1. Afterwards, the approach followed in our procedure is presented in section 1.3.2.

1.3.1. The optimization approach: a poor breakthrough potential

According to [8,33], optimization represents the most widespread design approach in the industrial innovation environment. Its use is enormously increased thanks to computer diffusion. As introduced in section 1.1.2.2, the optimization generally consists of obtaining a definitive design layout, whose parameters are optimized according to the given specifications (cf. Fig. 4). Section 1.3.1.1 provides the typical flowchart of an optimization procedure and some general definitions regarding this approach. Based on these definitions, an example of optimization is then proposed in sections 1.3.1.2 and 1.3.1.3. Finally, the aspects of the optimization approach which collide with our purposes and common targets (cf. Table 9) are pointed out and analysed in section 1.3.1.4.

1.3.1.1. Optimization flowchart and general definitions

The typical flowchart of an optimization procedure is reported in Fig. 12. Based on Fig. 2 and Fig. 4, an initial concept variant (an engineering structure, cf. Fig. 4) and some specifications are assumed as start points. Regarding the annotations used in Fig. 12:

- "v" (lowercase) indicates the generic single variable, while "V" (capital) indicates the entire set of variables.
- "p" (lowercase) indicates the generic single performance, while "P" (capital) indicates the entire set of performances.
- The subscripts "n" and "m" respectively represent the total number of variables and the total number of performances.
- The subscript "obj" indicates an objective (target) to achieve.
- The subscript "i" indicates the ith generic iteration of the optimization process.

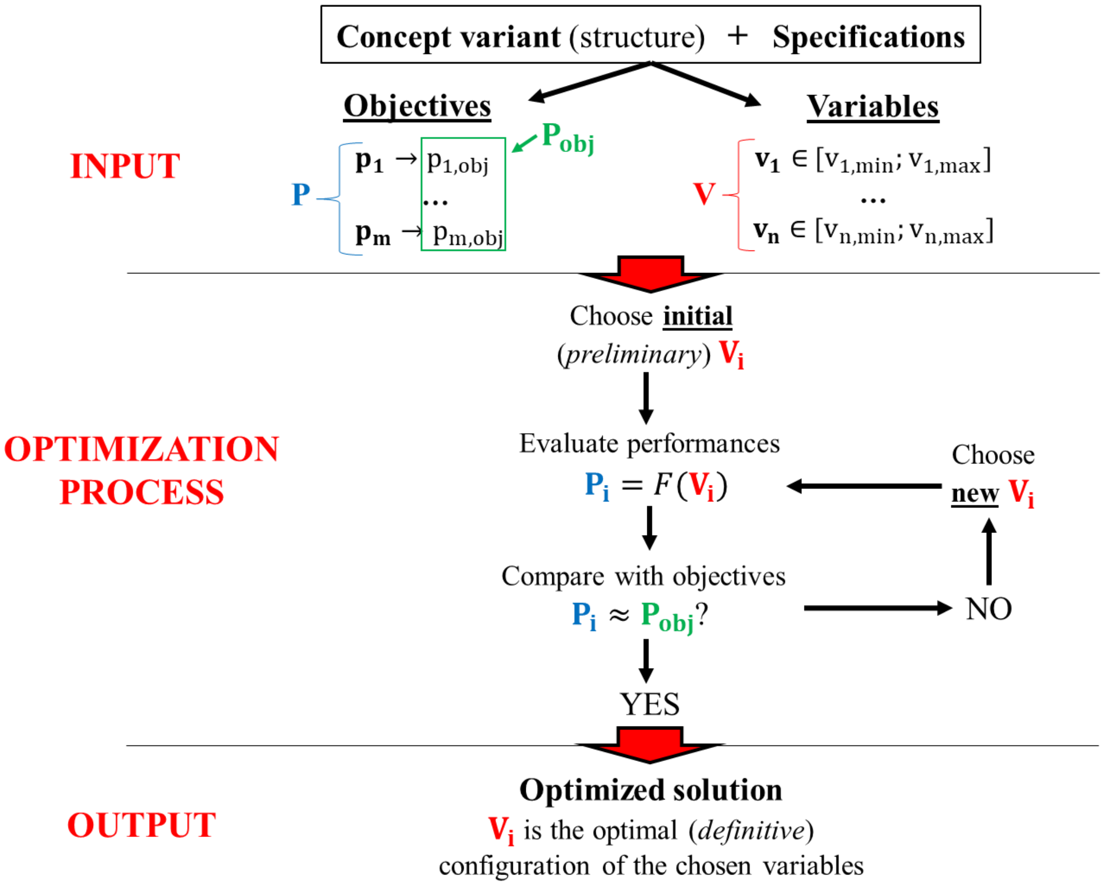

Concept variant (structure) + Specifications
Objectives
Variables
INPUT
P
$p_1 \rightarrow p_{1,obj}$
...
$p_m \rightarrow p_{m,obj}$
P_{obj}
V
$v_1 \in [v_{1,min}; v_{1,max}]$
...
$v_n \in [v_{n,min}; v_{n,max}]$
OPTIMIZATION PROCESS
Choose initial (preliminary) V_i
Evaluate performances $P_i = F(V_i)$
Choose new V_i
Compare with objectives $P_i \approx P_{obj}$?
NO
YES
OUTPUT
Optimized solution
V_i is the optimal (definitive) configuration of the chosen variables

Fig. 12: Typical flowchart of an optimization procedure (cf. [8]).

The optimization approach is characterised by three main phases: input, optimization process and output.

Input. This phase concerns the definition of two main groups of parameters:

- Objectives of the optimization, i.e. the performances which the optimized (definitive) structure must achieve. The performance parameters (p) and the corresponding target values (p_{obj}) are defined based on the required specifications. The objectives can generally concern different characteristics of the studied structure: mechanical (as mass, cf. Fig. 4, structural stiffness, etc.), economic (as material price, production price, etc.), etc. We define the optimization as multi-objective if the objectives are more than one [1,8].
- Variables, i.e. the design parameters of the studied structure which have to be optimized according to the required objectives. Each variable is defined in a limited variation range ($[v_{min} ; v_{max}]$, cf. Fig. 12).

Optimization process. The iterative process of optimization constitutes the core of the optimization approach. The choice of an initial configuration of the design variables (initial V_i, cf. Fig. 12) is required to start the process: a preliminary value is assumed for each variable (preliminary layout, cf. Fig. 4). Hence, the set of performances P_i is evaluated as function of V_i. Notably, *F* (cf. Fig. 12) represents the ensemble of functions describing the behaviour of the i^{th} design layout. Finally, P_i is compared with P_{obj} to verify if the required objectives have been achieved. If yes, the optimization process is ended; otherwise, the optimization process continues with a new iteration, by choosing a new configuration of the design variables (new V_i, cf. Fig. 12), different from the previous one. In general, the iterative process shown in Fig. 12 is automatically handled with advanced informatic procedures, as gradient and genetic algorithms (cf. [40,41]), which also contribute to speed up the optimization process.

Output. Once the optimization process is ended, the last V_i constitutes the optimal configuration of the chosen design variables (i.e. the definitive design layout of the studied structure, cf. Fig. 2 and Fig. 4).

1.3.1.2. An example of multi-objective parametric optimization

We propose an example of multi-objective parametric optimization. Based on Fig. 12, the input phase of the proposed example is reported in Fig. 13. The studied concept variant involves a cantilever beam with a squared section. The hypothesis of isotropic material is also assumed. As suggested by the legend, the design parameters linked to the studied structure can be distinguished between:

- Potential variables (highlighted in red, cf. Fig. 13), i.e. parameters which can potentially become design variables, and thus be optimized. In our example, the section side (h), the Young's modulus (E) and the material density (ρ) belong to this category.
- Work conditions (highlighted in green, cf. Fig. 13), i.e. the parameters which describes the work environment of the concept variant and are therefore priorly fixed due to external exigences. In our example, the extremity load (F) and the beam length (L) belong to this category.

The specifications are finally reported on the right side of Fig. 13.

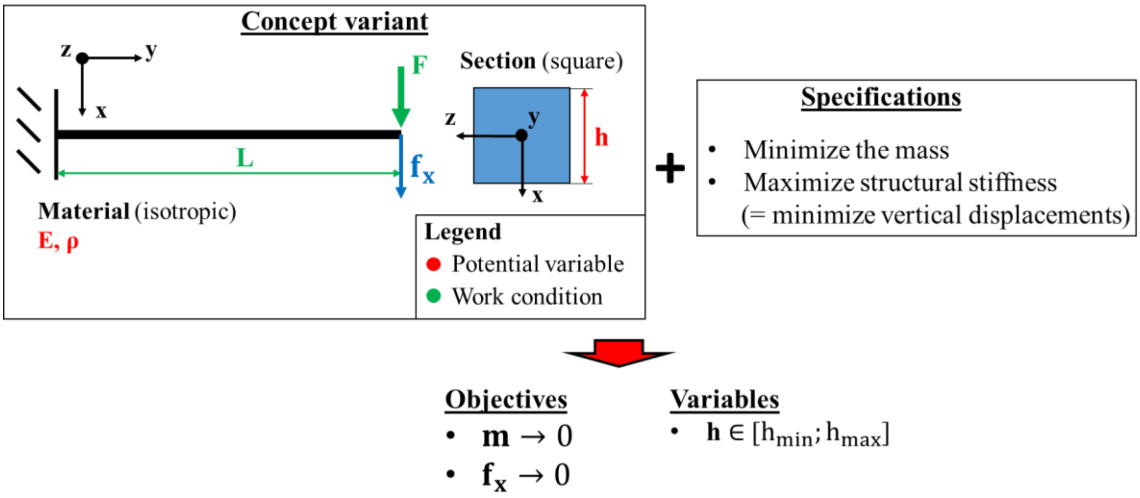


Fig. 13: Input phase of the proposed example (according to Fig. 12).

Based on the required specifications, the mass of the beam (m) and the vertical displacement at the extremity (f_x) represent our two performances. Since no target value is indicated in the specifications, m_{obj} and $f_{x,obj}$, are assumed equal to zero (i.e. m and f_x must be minimized as much as possible). We are thus in a multi-objective optimization since we have more than one objective. For brevity, the section side (h) represents the unique variable assumed, whose value is limited in the variation range [h_{min} ; h_{max}], while the other potential variables (E and ρ) are considered constant (cf. Fig. 13). The assumed values of the design parameters are reported in Table 10.

Table 10: Assumed values of the design parameters in the example of Fig. 13.

Parameter	Value and measure unit
h	[10; 30] mm
E	210 000 MPa
ρ	7800 kg/m^3
F	10 N
L	1000 mm

Choosing only one variable enables to simplify the optimization process (cf. Fig. 12): we can directly study the dependency of m and f_x on h, without using any iterative procedure. Hence, Eqs. (1) and (2) describe the behaviour of the performances m and f_x as function of the variable h (*F*, cf. Fig. 12). Eq. (2) has been adopted by assuming the hypothesis of linear elasticity, according to [42].

$$m = \rho L h^2 \tag{1}$$

$$f_x = \frac{FL^3}{3E\left(h^4/_{12}\right)} \tag{2}$$

The relationship m-f_x in Fig. 14a has been obtained with Eqs. (1) and (2), by varying h between 10 and 30 mm (cf. Table 10), and it represents the ensemble of possible solutions. Notably, each point of the curve is characterised by different values of h and of the corresponding performances. It can be remarked from Fig. 14a that the performances m and f_x are in contradiction[2]: an improvement in the mass (decreasing of m, cf. Fig. 13), obtained by decreasing h, leads to a deterioration of the vertical stiffness (increasing of f_x). This means the optimal solution, i.e. the optimized value of the variable h (cf. Fig. 12), will be a compromise between the performances m and f_x.

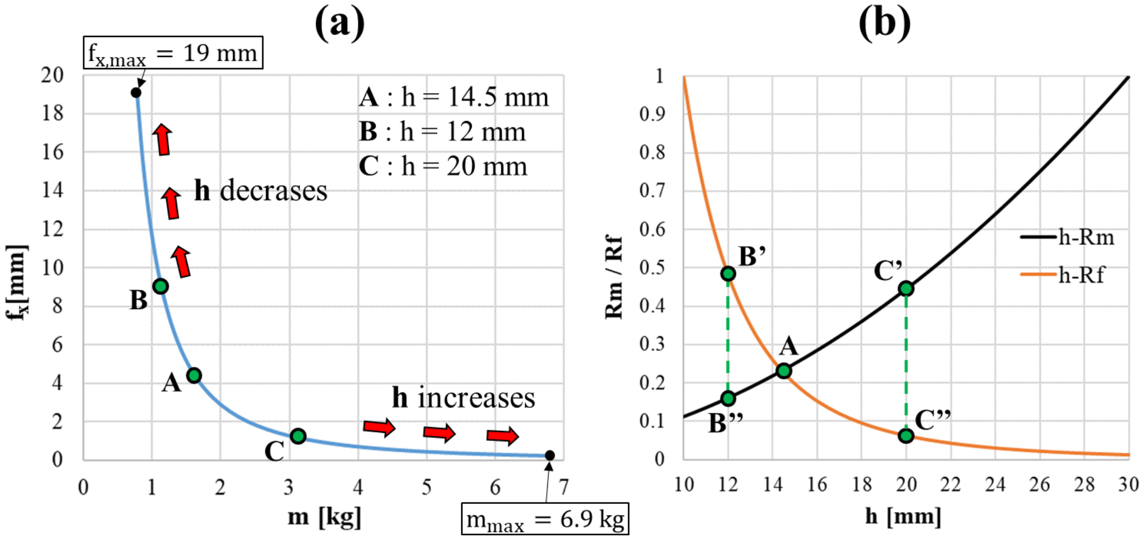


Fig. 14: (a) Relationship m-f_x (b) Curves h-Rm and h-Rf.

The curves h-R_m and h-R_f in Fig. 14b can help to select the "best" compromise solution. Notably, according to Eqs. (3) and (4) (cf. [40,41]), R_m and R_f correspond to the performances m and f_x (cf. Eqs. (1) and (2)) normalized between the objectives m_{obj} and $f_{x,obj}$ (cf. Fig. 13) and the respective maximum values m_{max} and $f_{x,\ max}$ (cf. Fig. 14a). Therefore, the curves h-R_m and h-R_f represent the trend of the normalized performances in the variation range of h ([10; 30] mm, cf. Table 10).

$$R_m = \frac{m\ - m_{obj}}{m_{max} - m_{obj}} \tag{3}$$

$$R_f = \frac{f_x - f_{x,obj}}{f_{x,max} - f_{x,obj}} \tag{4}$$

Clearly, as shown in Fig. 14b, the minimum mass ($R_m \approx 0.11$) is obtained with h = 10 mm, while the minimum vertical displacement ($R_f \approx 0.02$) is obtained with h = 30 mm. The points A, B and C in Fig. 14 represent three different compromise solutions:

[2] The term "contradiction" indicates a condition in which a certain performance parameter cannot be improved without deteriorating another one [11]. According to the example in Fig. 13, reducing the mass of the beam (improvement) by decreasing the section side h makes the vertical displacement increasing (deterioration).

A) h = 14.5 mm (m = 1.6 kg, f_x = 4.3 mm, cf. Fig. 14a). This solution is given by the intersection of the curves h-R_m and h-R_f. According to Fig. 14b, the point A corresponds to $R_m \approx R_f \approx 0.23$ (the "distance" m-m_{obj} is equal to the "distance" f_x-$f_{x,obj}$). In this case, m and f_x have the same weight.

B) h = 12 mm (m = 1.1 kg, f_x = 9.2 mm, cf. Fig. 14a). According to B'B'' (cf. Fig. 14b), this solution corresponds to $R_m \approx 0.16$ and $R_f \approx 0.48$. In this case, the minimization of m has the priority over the minimization of f_x.

C) h = 20 mm (m = 3.1 kg, f_x = 1.2 mm, cf. Fig. 14a). According to C'C'' (cf. Fig. 14b), this solution corresponds to $R_m \approx 0.44$ and $R_f \approx 0.06$. In this case, the minimization of f_x has the priority over the minimization of m.

Depending on the purpose of the optimization, all these compromises can be suitable optimal solutions. Since no priority between m and f_x was indicated in the required specifications (cf. Fig. 13), the compromise A (h = 14.5 mm, cf. Fig. 14a) is assumed as the optimized solution. In general, as shown in this example, a multi-objective optimization requires the definition of "priorities" (weights, assumed by designer) among the performance parameters in contradiction. This choice determines the nature of the compromise between the performances, and thus the form of the optimized solution [8].

1.3.1.3. Overcoming performance contradictions by means of conceptual variations

As argued in [8], some contradictions between performances can be overcome by means of conceptual variations, i.e. qualitative changes of the studied concept variant (cf. Fig. 13). The theoretical effect of qualitative changes on the optimization process is illustrated in Fig. 15a, adapted from [8], where the interdependency between two generic conflicting performances A and B is schematically represented with an hyperbole.

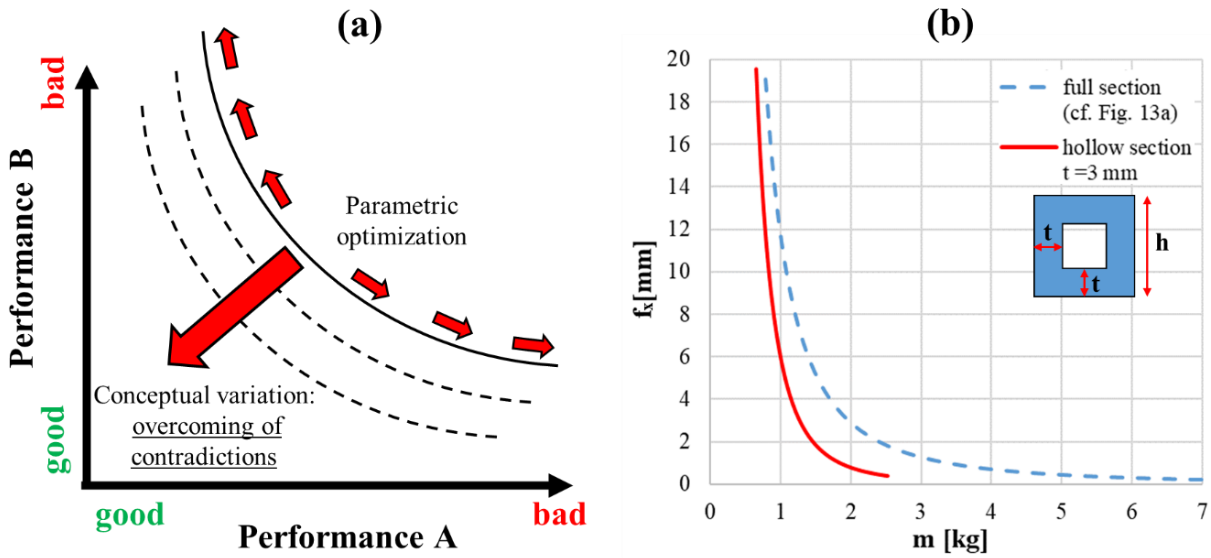


Fig. 15: (a) Parametric optimization vs conceptual variation, theoretical representation (cf. [8]) ; (b) Full squared section (cf. Fig. 13) vs hollow squared section with constant thickness.

As shown in Fig. 15a, a traditional parametric optimization "moves along the hyperbole" (cf. Fig. 14a), searching for the best compromise in the contradiction between the performances A and B. Instead, the introduction of a conceptual variation enables to achieve further improvements by overcoming the contradictions between A and B, i.e. "moving among the hyperboles" [8]. This behaviour can be easily remarked by introducing a hollow squared section, i.e. a qualitative change of the beam section shape, in the concept variant of Fig. 13, and observing its effect on the relationship between the performances m and f_x. In Fig. 15b, the blue dashed curve refers to the full squared section (cf. Fig. 14a, Eqs. (1) and

(2)), while the red curve refers to a hollow squared section with a constant thickness of 3 mm. Notably, the red curve has been obtained with Eq. (5) and (6) (cf. [42]), by varying h between 10 and 30 mm (cf. Table 10). According to [42], Eqs. (1) and (2) can be obtained from equation Eq. (5) and (6) respectively, by imposing t = h/2 (full squared section, cf. Fig. 13).

$$\mathrm{m} = \rho\mathrm{L}(\mathrm{h}^2 - (\mathrm{h} - 2\mathrm{t})^2) \tag{5}$$

$$\mathrm{f} = \frac{\mathrm{FL}^3}{3\mathrm{E}\left(\mathrm{h}^4/_{12} - {(\mathrm{h} - 2\mathrm{t})^4}/_{12}\right)} \tag{6}$$

According to the theoretical representation in Fig. 15a, the qualitative change of the section shape shifts the relationship m-f_x towards the left side (cf. Fig. 15b). Indeed, this enables to achieve solutions which are more optimized (m and f_x can be further minimized) than the compromises A, B and C previously obtained (cf. Fig. 14). Moreover, the parameter t can be adopted as design variable, allowing to test more combinations and thus widening the solution space. On the other hand, the adoption of a hollow squared section results in a new contradiction, now represented by the red curve in Fig. 15b. In general, this means that further improvements can be obtained by adopting further different qualitative changes on the concept variant [8]. However, this example demonstrates how conceptual variations can be a very powerful design tool.

1.3.1.4. The aspects of the optimization approach which collide with our purposes and targets

Based on the example proposed in sections 1.3.1.2 and 1.3.1.3, we report and analyse in this section the aspects of the optimization approach which collide with our purposes and common targets (cf. Table 9), denoted CAs (Colliding Aspects). We identified four main CAs:

CA 1. The optimization itself is out of our scope. According to Fig. 2, the optimization is typically included in the phase of embodiment design, when we are instead focused on conceptual design. In general, designers tend to skip the phase of conceptual design, by prematurely adopting the optimization approach and focusing on very few concept variants. As a result, the solution space is immediately restricted: the solution research is constrained to the assumed concept variants, while different and potentially interesting conceptual solutions are excluded a priori [18]. On the contrary, we want to enlarge the solution space as much as possible, by generating multiple and different concept variants (cf. Table 9).

CA 2. Too much focus on technical aspects. As seen in section 1.3.1.2, the optimization approach requires the definition of specific design parameters (potential variables and work conditions, cf. Fig. 13) and behaviour models (cf. Eqs. (1), (2), (5) and (6)), needed to describe the technical dimension of the studied concept variant. Notably, the research of the solution is constrained to the chosen behaviour models and is localised in the ranges of the chosen variables. This contributes to further restrict the solution space [1,8]. On the contrary, as reported in Table 9, we want to be as free as possible from technical aspects, to better favour the birth of unconventional conceptual solutions (cf. Table 9).

CA 3. Optimal solutions. Solutions are ranked according to the optimization objectives: all the solutions which do not satisfy the required objectives (i.e. the "bad" solutions) are usually discarded, contrary to the indications provided by the BDPs (cf. Table 5). Moreover, when multiple objectives are involved (cf. Fig. 13), the optimization approach leads to compromise solutions, which prevent to achieve further improvements. According to Fig. 15, better compromises, and thus more optimized solutions, can be obtained by adopting suitable conceptual variations [8].

CA 4. Many choices by designer are required. The required choices mostly regard the preliminary numerical values, which must be assumed by designer before starting the optimization process. As seen in section 1.3.1.2, the preliminary values include the limits of the variable ranges and the other constant design parameters (cf. Table 10), needed to use the chosen behaviour models (cf. Eqs. (1) and (2)). In case of multi-objective optimization, the definition of priorities among the performances in contradiction may be also required. The more choices by designer are required, the more cognitive biases risk to affect the design process.

The analysed CAs are summarised in Table 11. Notably, Table 11 helps to better highlight the reasons why the optimization approach is not suitable to our purposes (cf. Table 9). It can be remarked that many characteristics of the analysed CAs can be reconducted to some of the BDHs reported in Table 7, as the excessive focus on technical aspects and on optimal solutions. We believe the optimization approach could be very useful to reach incremental innovations, based on an experienced knowledge of the assumed design variables and performances. However, it may not be suitable to help designer in searching for original design concepts and overcoming his cognitive biases [2].

Table 11: Aspects of the optimization approach colliding with our purposes (cf. Table 9).

CA	What it involves	What instead we want to do
1. Out of scope	The solution research is constrained to one ore few concept variants	Focus on conceptual design to generate different concept variants
2. Too much focus on technical aspects	The solution research is constrained to the chosen behaviour models and is localised in the range of the chosen variables	Free designer from technical aspects to: • Enlarge the solution space • Allow unconventional solution paths
3. Optimal solutions	All the solutions which do not satisfy the required objectives (i.e. bad solutions) are discarded	Insist on the BDPs (cf. Table 5): search for multiple and different solutions by keeping also bad solutions
4. Many choices by designer are required	More risk of influence of the cognitive biases	Prevent cognitive biases to affect the designer’s choices

1.3.2. A natural language-based approach for conceptual design

In the current section, the approach followed in our work is presented. According to our purposes (cf. Table 9), we thus have to focus on conceptual design. Based on Fig. 2 and Fig. 3, the two main steps of the conceptual design phase are shown in Fig. 16 [11,16].

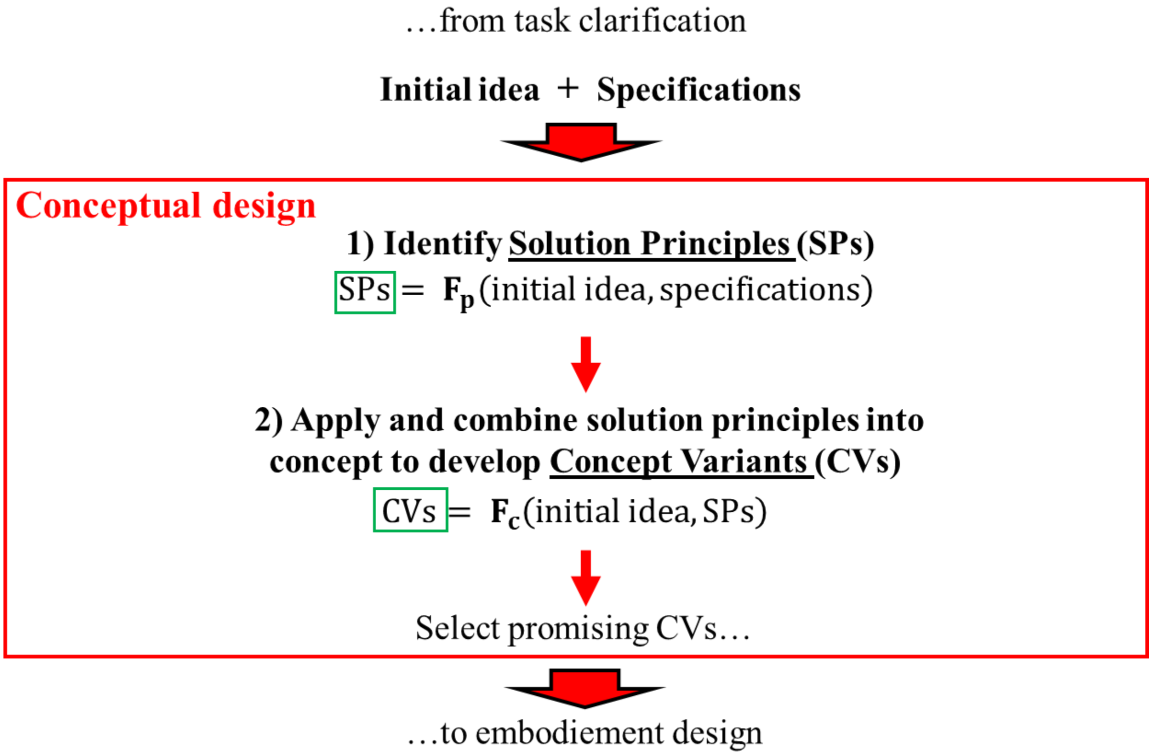

…from task clarification
Initial idea + Specifications
Conceptual design
1) Identify Solution Principles (SPs)
SPs = $\mathbf{F}_p$(initial idea, specifications)
2) Apply and combine solution principles into concept to develop Concept Variants (CVs)
CVs = $\mathbf{F}_c$(initial idea, SPs)
Select promising CVs…
…to embodiement design

Fig. 16: Main steps of the conceptual design phase.

The first step concerns the identification of the solutions principles, denoted SPs, as function of the initial idea and of the required specifications, while the second step involves the application of the identified solution principles to the initial idea, for generating the concept variants, denoted CVs. For now, we do not focus on the evaluation and selection of the promising CVs, since our purpose is to be able to generate as many different CVs as possible (cf. Table 9). Based on section 1.1.2.1, SPs and CVs can be defined as follows:

- **SP**: general strategy to satisfy the required design specification (cf. Fig. 3).
- **CV**: representation expressing a different declination of the design concept (cf. Fig. 3).

The functions F_p and F_c generically represent the methods to realize the steps 1 and 2, respectively (cf. Fig. 16). The detailed description of the methods implemented in our aided design procedure is addressed in the next chapter. Here, instead, we want to introduce the fundamental principles which characterise the followed approach, denoted PAAs (Principles of the Adopted Approach), i.e. the way we want to face with conceptual design. Notably, the study of a well-established conceptual design theory, called TIPS (Theory of Inventive Problem Solving), will enable to better introduce the PAAs. The current section is organized as follows. In section 1.3.2.1 we present TIPS, analysing the conceptual design procedure provided by this theory. Hence, an example of the application of TIPS, reported in literature, is illustrated in section 1.3.2.2. Finally, the PAAs are introduced and discussed in section 1.3.2.3.

1.3.2.1. The Theory of Inventive Problem Solving (TIPS)

As seen in the example proposed in section 1.3.1.2 (cf. Fig. 13), design specifications are typically characterised by contradictions[2] among the required objectives. In general, this leads to compromise solutions (cf. Fig. 14) and prevents to achieve further improvements during an optimization process [8,16]. However, the benefit involved by conceptual variations in overcoming the contradiction between the objectives has been analytically demonstrated in section 1.3.1.3 (cf. Fig. 15). Based on these ideas, the Russian engineer and inventor G. Altshuller formulated and developed the Theory of Inventive Problem-Solving (TIPS, also known as TRIZ from the Russian abbreviation, cf. [43,44]), from 1946 to 1988. Altshuller and his colleagues analysed a large number of patents and observed that most of them suggested means for eliminating the specification contradictions in a system. Notably, these means were based on recurrent solution (or inventive) principles and used recurrent geometrical, physical and chemical effects to realize the system functions. This allowed Altshuller to synthetise 40 general SPs[3] and about 2500 effects, and to finally organize them in a conceptual design theory (i.e. TIPS) [43,44]. Based on Fig. 16, the conceptual design procedure provided by TIPS, adapted from [11,16], is schematically represented in Fig. 17. For a better comprehension of the procedure, each step has been decomposed according to inputs, used methods and outputs.

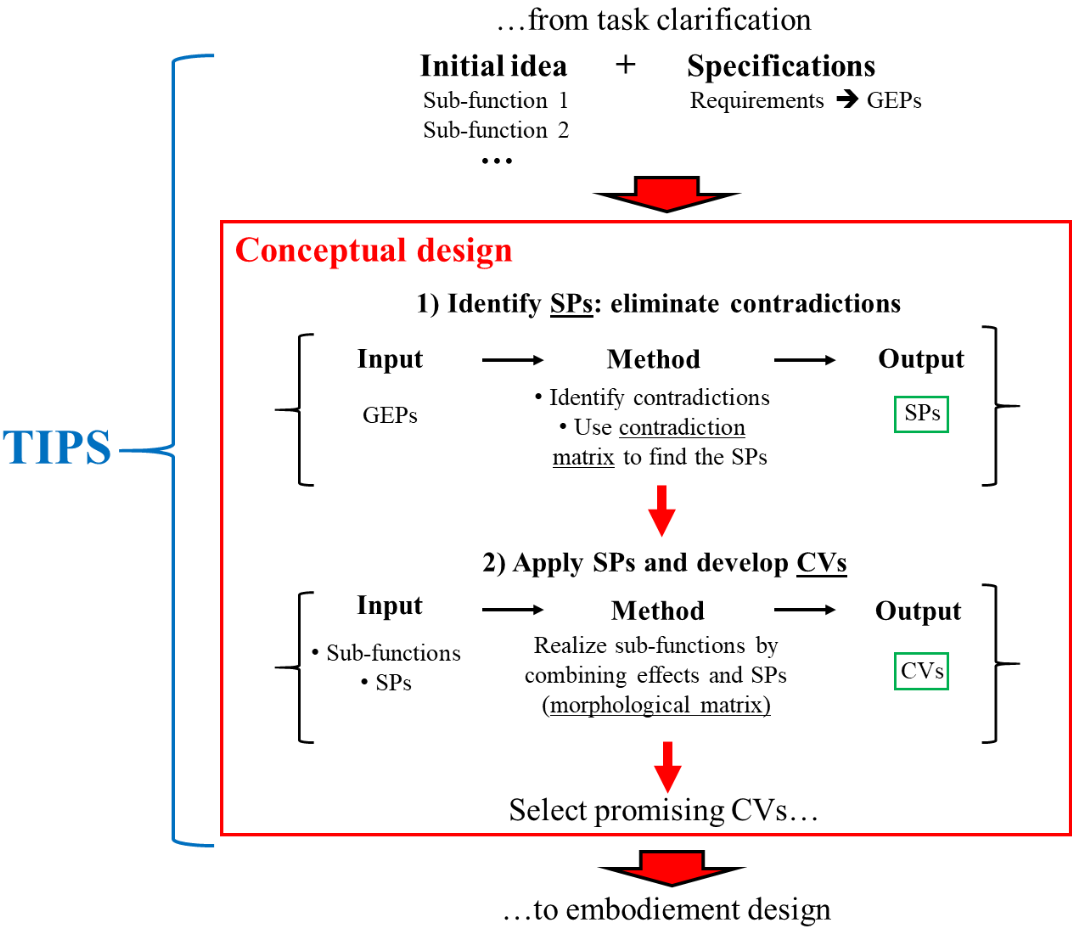


Fig. 17: Conceptual design procedure provided by TIPS (cf. [11,16,45]).

[3] Complete list of the 40 principles (indications and examples): http://www.triz40.com/aff_Principles_TRIZ.php

As indicated in Fig. 17, the initial idea must be defined in terms of sub-functions, whose sequence describes the functioning of the studied system. According to Fig. 3, the specification requirements are expressed as generalised engineering parameters, denoted GEPs (cf. Fig. 17), to be improved during conceptual design. TIPS provides 39 generalised parameters which must be used by designer to express the design specifications [43,44]. We can now analyse the two main steps of the procedure.

Step 1. The SPs are identified by eliminating the contradictions among the GEPs. Notably, the pairs of contradicting GEPs are fed into contradiction matrix, which provides the suitable SPs to eliminate the contradictions (cf. Fig. 17). A reduced section of the contradiction matrix is reported in Table 12, according to [11,43]. The rows and the columns of the matrix contain the 39 generalised parameters[4] made available by TIPS. The numbers in the matrix cells identify the SPs which allow to eliminate a contradiction between two given GEPs. The empty cells indicate the pairs of not conflicting GEPs, while the grey cells indicate the matrix diagonal.

Table 12: Section of the contradiction matrix (cf. [11,43]).

			GEPs which deteriorate				
			1	**2**	**…**	**14**	**…**
			Weight of stationary object	Length of moving object	**…**	Strength	**…**
GEPs to be improved	**1**	Weight of stationary object			**…**	2, 10, 27, 28	**…**
	2	Length of moving object			**…**	8, 29, 34, 35	**…**
	…	**…**	**…**	**…**	**…**	**…**	**…**
	14	Strength	1, 26, 27, 40	1, 8, 15, 35	**…**		**…**
	…	**…**	**…**	**…**	**…**	**…**	

For example, suppose we need to resolve the contradiction between the GEPs "strength" and "weight of stationary object". Using the contradiction matrix in Table 12, the following SPs are suggested (cf. [43,44]):

- 1: segmentation.
- 26: copying.
- 27: cheap short-living objects.
- 40: composite materials.

As argued in [11], the 40 SPs[3] provided by TIPS do not constitute final solutions, but rather high-level general strategies for finding ideas on how to exploit physical phenomena in various contexts. Indeed,

[4] Visualize the entire contradiction matrix:
- Classical contradiction matrix (39 GEPs, cf. [43]): http://www.triz40.com/aff_Matrix_TRIZ.php
- A more recent version (48 GEPs): https://triz-journal.com/contradiction-matrix/

they are often supplemented with additional indications and design examples in the operational implemented versions of TIPS.

Step 2. Generating the CVs involves the realization of the system sub-functions (cf. Fig. 17). This can be done by building a morphological matrix which allows to combine the found SPs with the effects provided by TIPS. Notably, each effect expresses a physical, mechanical or chemical working principle, and is associated to a specific design case, i.e. an illustrated example of the involved effect. An example of morphological matrix is reported in Table 13 (cf. [18,45]), where "n" indicates the number of subfunctions and "m" the number of solutions, and in which we supposed to have 4 SPs found in the previous step.

Table 13: Example of morphological matrix of effects and design cases (cf. [18,45]).

		Solutions			
		1	2	**…**	m
Sequence of the system sub-functions	Sub-function 1	Effect 1.1 Design case 1.1 SP 2	Effect 1.2 Design case 1.2 SP 2 + SP 3	…	
	Sub-function 2	Effect 2.1 Design case 2.1 SP 1	Effect 2.2 Design case 2.2 SP 3	…	Effect 2.m Design case 2.m SP 2
	…	…	…	…	…
	Sub-function n	Effect n.1 Design case n.1 SP 1 + SP 4	Effect n.2 Design case n.2 SP 4	…	

Each case contains an effect and a corresponding design case. Effects and design cases are identified with two numbers, indicating the sub-function (row) and the solution (column), respectively. As shown in Table 13, the effects are used to contemporarily realize a given sub-function of the system and to apply one or more of the found SPs. For instance, the sub-function 1 can be realized with the effect 1.1 (design case 1.1), which involves the application of the SP 2. The CVs can be finally generated with the morphological matrix, by realizing all the sub-functions required to the system, in sequence. For example, based on Table 13, three different CVs can be generated:

- **CV 1**: 1.1 - 2.2 - … - n.1
- **CV 2**: 1.2 - 2.1 - … - n.2
- **CV 3**: 1.2 - 2.m - … - n.1

Observing Table 13, it can be easily remarked that all the generated CVs involve the application of all the 4 SPs supposed. Depending on the complexity of the studied system, the construction of the morphological matrix by designer can be very challenging and time consuming. Indeed, TIPS provides a library of 30 basic functions, linked to the corresponding effects and design cases, to help designer building the morphological matrix. A schematical representation of the function library is reported in Fig. 18 (cf. [11]). A complete list of the 30 basic functions can be found in [11,43]. According to Fig. 18, some functions can have more associated effects than other functions, and this is the reason why some cells in the morphological matrix can be empty.

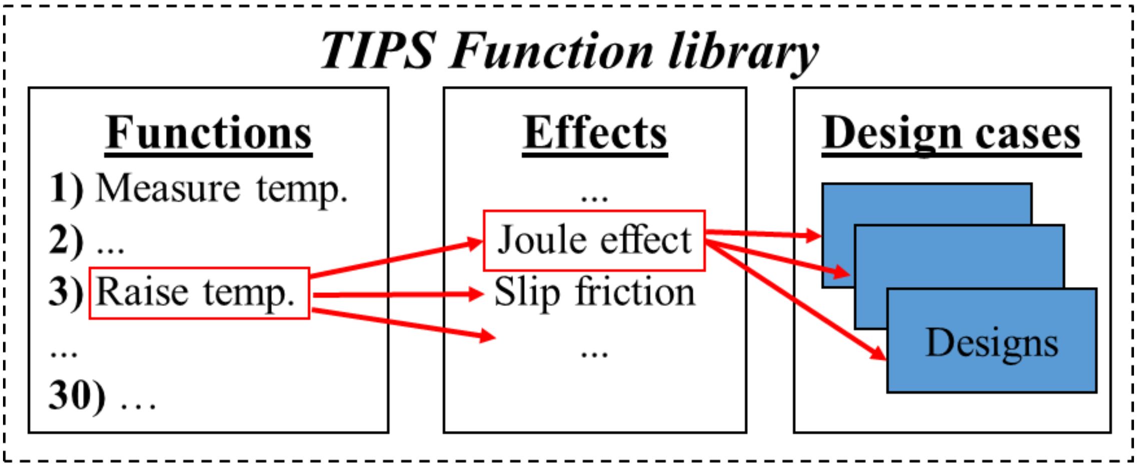


Fig. 18: Schematic representation of the function library provided by TIPS (cf. [11]).

The practical example proposed in section 1.3.2.2 can help to better understand the conceptual design procedure proposed by TIPS (cf. Fig. 17).

1.3.2.2. An example of the application of TIPS: the conceptual design of a paper punch system

The proposed example, concerning the conceptual design of a paper punch system, is reported in [18,45] and follows the procedure presented in section 1.3.2.1. According to Fig. 17, Fig. 19a reports the initial idea with the list of the system sub-functions, in sequence, while Fig. 19b shows the design requirements to improve, which have been synthetised by costumers' needs. As done in [45], the 9 requirements are reduced into 6 GEPs, chosen among the 39 parameters provided by TIPS, where the abbreviation "st." stands for "stationary object" [43]. The contradictions identified among the considered GEPs are reported for brevity on the right side of Fig. 19b [18].

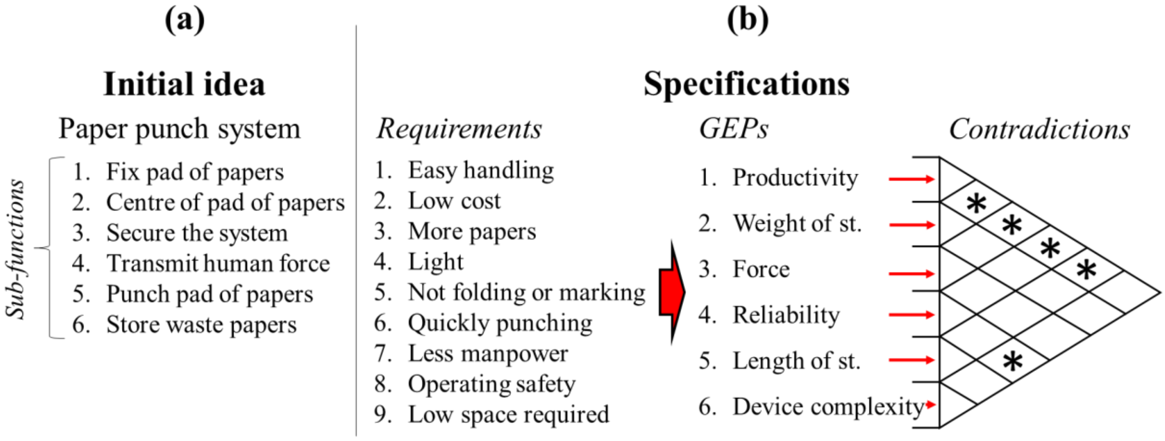
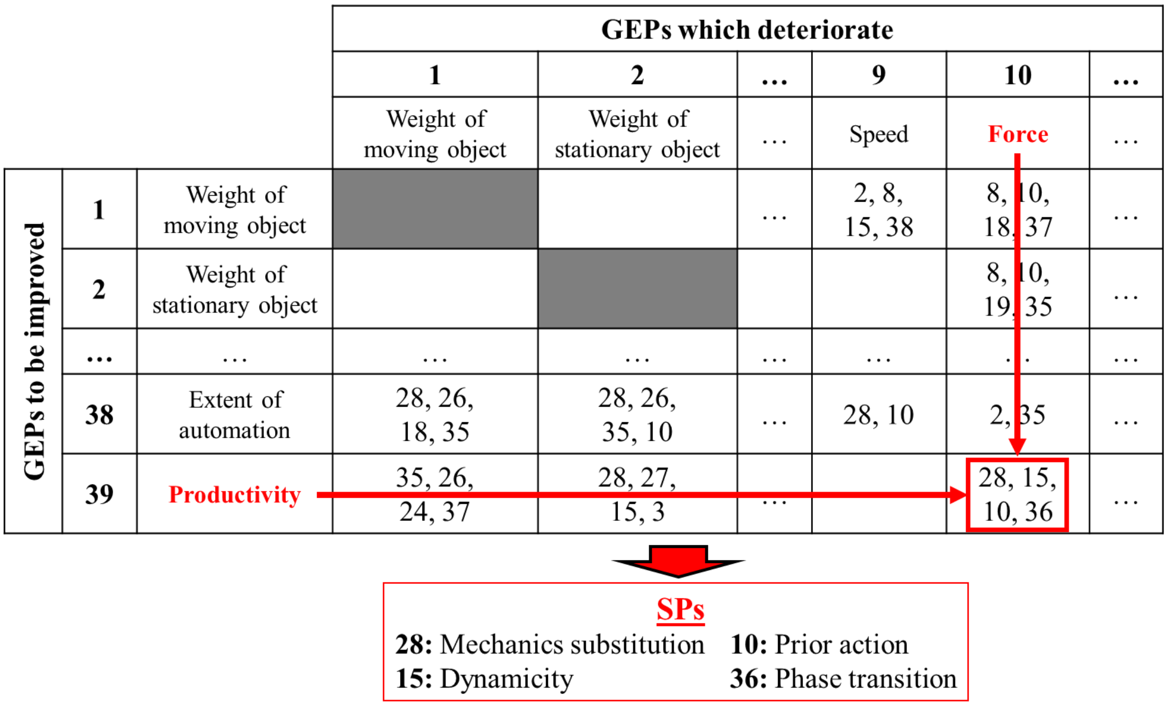


Fig. 19: (a) initial idea and related sub-functions; (b) design requirements to GEPs and identified contradictions (cf. [18,45]).

Step 1 (cf. Fig. 17). We have 5 contradictions (1-2, 1-3, 1-4, 1-5 and 4-6, cf. Fig. 19b) which must be eliminated with the contradiction matrix. Fig. 20 shows the use of the matrix to solve the contradiction productivity-force (1-3, cf. Fig. 19b), and the resulting SPs[3].



			GEPs which deteriorate					
			1	2	…	9	10	…
			Weight of moving object	Weight of stationary object	…	Speed	Force	…
GEPs to be improved	1	Weight of moving object	▨		…	2, 8, 15, 38	8, 10, 18, 37	…
	2	Weight of stationary object		▨			8, 10, 19, 35	…
	…	…	…	…	…	…	…	…
	38	Extent of automation	28, 26, 18, 35	28, 26, 35, 10	…	28, 10	2, 35	…
	39	Productivity	35, 26, 24, 37	28, 27, 15, 3	…		28, 15, 10, 36	…

SPs

28: Mechanics substitution **10:** Prior action
15: Dynamicity **36:** Phase transition

Fig. 20: Eliminating the contradiction productivity-force (1-3, cf. Fig. 19b).

As anticipated in section 1.3.2.1, TIPS provides further additional indications on how applying the SPs and some corresponding examples. For example, Table 14 (cf. [18,43,45]) reports the indications and the corresponding examples regarding the SP 28 (mechanics substitution[3], cf. Fig. 20).

Table 14: Additional indications and corresponding examples for the SP 28 (mechanics substitution, cf. Fig. 20).

Indication (how to apply the SP)	Corresponding example
Replace a mechanical means with a sensory (optical, acoustic, taste or smell) means.	• Replace a physical fence to confine a dog or cat with an acoustic "fence" (signal audible to the animal). • Use a bad smelling compound in natural gas to alert users to leakage, instead of a mechanical or electrical sensor.
Use electric, magnetic and electromagnetic fields to interact with the object.	• To mix 2 powders, electrostatically charge one positive and the other negative. Either use fields to direct them, or mix them mechanically and let their acquired fields cause the grains of powder to pair up.
Change from static to movable fields, from unstructured fields to those having structure.	• Early communications used omnidirectional broadcasting. We now use antennas with very detailed structure of the pattern of radiation.
Use fields in conjunction with field-activated (e.g. ferromagnetic) particles.	• Heat a substance containing ferromagnetic material by using varying magnetic field. When the temperature exceeds the Curie point, the material becomes paramagnetic, and no longer absorbs heat.

According to Table 14, the SP 28 suggests replacing a mechanical means by exploiting different physical phenomena, as acoustics, electric and magnetic fields. In general, each SP provided by TIPS can be applied in different ways by exploiting phenomena of different nature [43,44]. The other contradictions are finally solved in a similar way. Table 15 (cf. [18,45]) resumes all the contradictions and the respective SPs identified with the contradiction matrix.

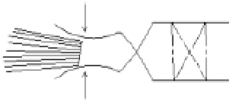
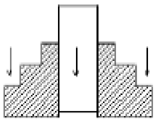
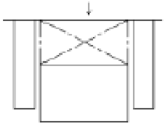
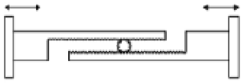
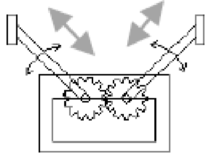
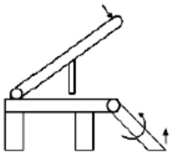
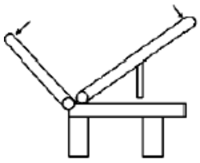
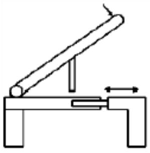
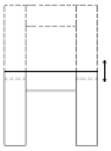
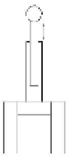
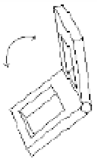
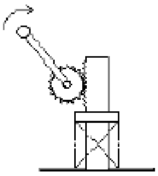
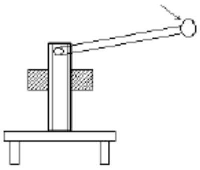
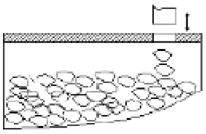
Table 15: Contradictions and respective identified SPs (cf. [18,45]).

Contradiction		IDs of the identified SPs	Selected SPs	
ID	Conflicting GEPs		ID	Name
1-2	Productivity-Weight of st.	28, 27, 15, 3	3	Local quality
1-3	Productivity-Force	28, 15, 10, 36	15	Dynamicity
1-4	Productivity-Reliability	1, 35, 10, 38	1, 10	Segmentation, Prior action
1-5	Productivity-Length of st.	30, 7, 14, 26	7, 14	Nesting, Spheroidality-curvature
4-6	Reliability-Device complexity	13, 35, 1	1	Segmentation

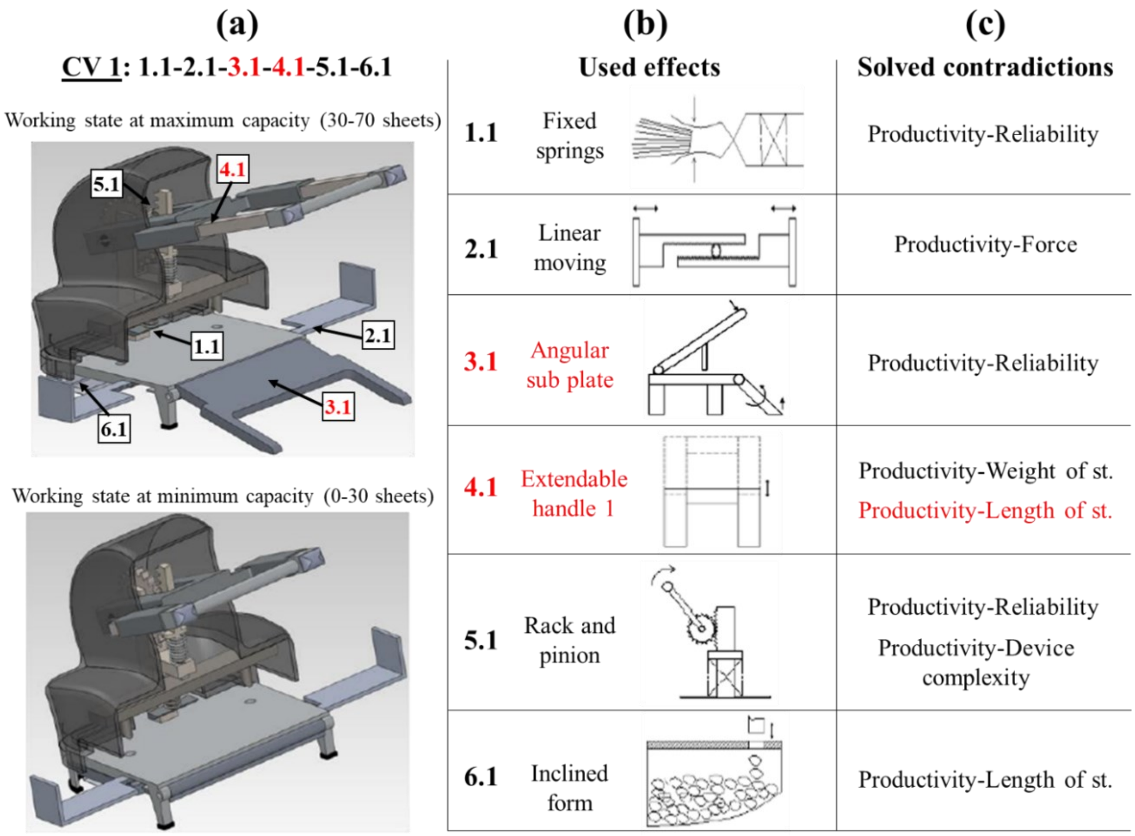
As shown in Table 15, few SPs for each contradiction are selected. According to [18,45], the selected SPs are more adequate to the current purpose and they also allow to solve more contradictions at once. For example, the SP 1 (segmentation[3]) can be applied to solve the contradictions 1-4 and 4-6, while the SP 15 (dynamicity[3]) to solve the contradictions 1-2 and 1-3.

Step 2 (cf. Fig. 17). We have now to build the morphological matrix of effects and design cases to realize the system subfunctions reported in Fig. 19a. Based on Table 13, the built morphological matrix is shown in Table 16. Each cell of the matrix reports an effect, with a corresponding design case, and the applied SPs, among the ones selected in Table 15. We can remark that all the selected SPs (cf. Table 15) are present in Table 16, at least one time. As seen in Table 13, some effects allow to apply two distinct SPs, as for example the “foldable handle”, which can be used to realize the sub-function “4. Transmit human force” and to apply the SPs “segmentation” and “local quality”, thus solving two contradictions, i.e. 1-4 and 1-2 respectively (cf. Table 15). Moreover, the same SP can be applied by means of different effects. For instance, according to Table 16, the SP “prior action” (contradiction 1-4, cf. Table 15) can be applied by means of the effects “fixed with spring”, “pre-pressure plate around punchers” and “pre-pressure plate between the punchers”, realizing the sub-function “1. Fix pad of papers”. The same for the sub-function “3. Secure the system”. We can also have different variants of the same effect which realize the same function, as for the sub-function “4. Transmit human force” which can be realized by means of two different variants of “Extendable handle”, thus involving two different design cases (cf. Fig. 18). Hence, the CVs can be now generated by combining the different effects reported in Table 16, to realize all the required sub-functions and to solve all the identified contradictions (cf. Table 15).

Table 16: Morphological matrix of effects, design cases and selected SPs, built for the paper punch system (cf. [18,45]).

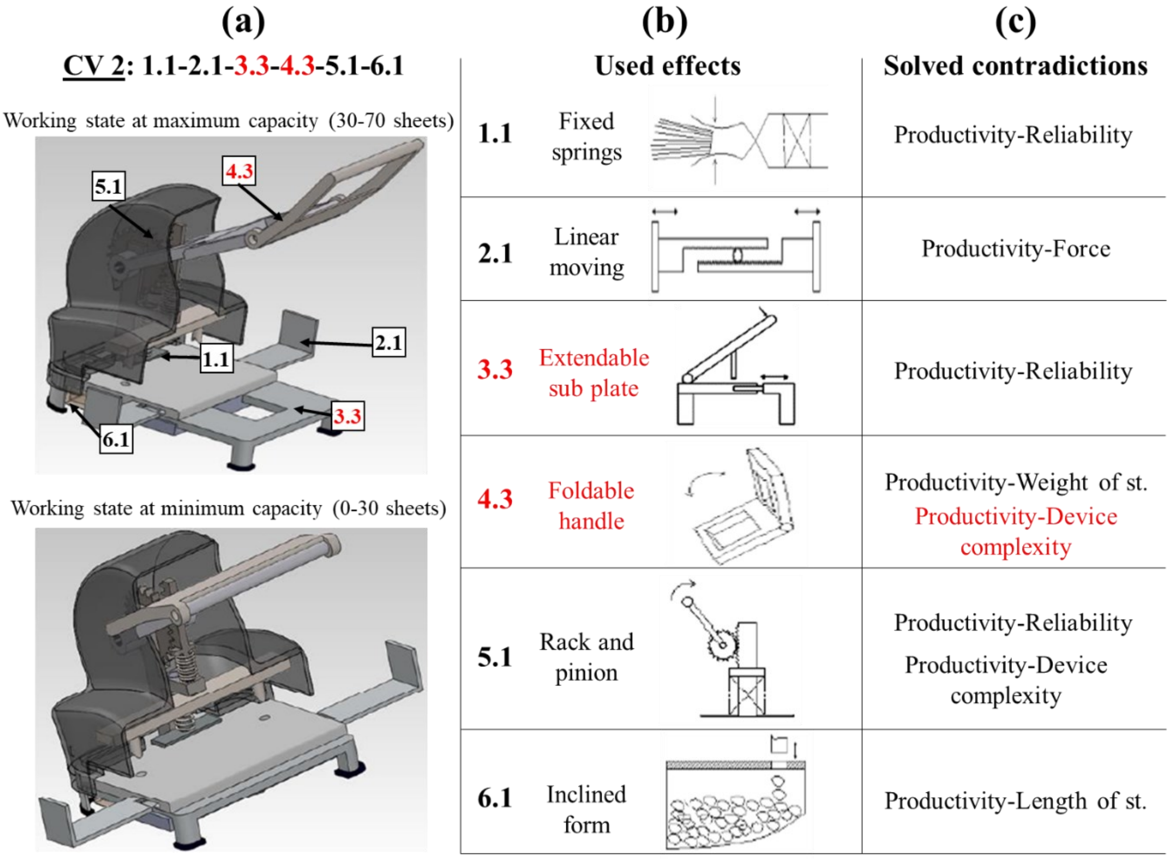
		Solutions		
		1	2	3
Sequence of the system sub-functions	1. Fix pad of papers	Fixed with springs  Prior action (10)	Pre-pressure plate around punchers  Prior action (10)	Pre-pressure plate between punchers  Prior action (10)
	2. Centre of pad of papers	Linear moving  Dynamicity (15)	Angular moving  Spheroidality (14) + dynamicity (15)	
	3. Secure the system	Angular sub plate  Prior action (10)	Reverse handle  Prior action (10)	Extendable sub plate  Prior action (10)
	4. Transmit human force	Extendable handle 1  Nesting (7) + Local quality (3)	Extendable handle 2  Nesting (7) + Local quality (3)	Foldable handle  Segmentation (1) + Local quality (3)
	5. Punch pad of papers	Rack and pinion  Segmentation (1)	Screw  Spheroidality (14)	
	6. Store waste papers	Inclined form  Spheroidality (14)		

Two examples of CVs, denoted CV 1 and CV 2, are reported in Fig. 21 and Fig. 22 according to [18,45]. Notably, Fig. 21a and Fig. 22a provide a geometry representation of CV 1 and CV 2, respectively, in the working states of maximum and minimum capacity. Fig. 21b and Fig. 22b illustrate the used effects, while Fig. 21c and Fig. 22c show the solved contradictions. The differences between CV 1 and CV 2 are highlighted in red (cf. Fig. 21 and Fig. 22).


(a)
CV 1: 1.1-2.1-3.1-4.1-5.1-6.1
Working state at maximum capacity (30-70 sheets)
5.1
4.1
2.1
1.1
3.1
6.1
Working state at minimum capacity (0-30 sheets)
(b)
(c)

Used effects		Solved contradictions
1.1	Fixed springs	Productivity-Reliability
2.1	Linear moving	Productivity-Force
3.1	Angular sub plate	Productivity-Reliability
4.1	Extendable handle 1	Productivity-Weight of st. Productivity-Length of st.
5.1	Rack and pinion	Productivity-Reliability Productivity-Device complexity
6.1	Inclined form	Productivity-Length of st.

Fig. 21: CV 1 : (a) geometry representation ; (b) used effects ; (c) solved contradictions (cf. [18,45]).


(a)
CV 2: 1.1-2.1-3.3-4.3-5.1-6.1
Working state at maximum capacity (30-70 sheets)
5.1
4.3
2.1
1.1
3.3
6.1
Working state at minimum capacity (0-30 sheets)
(b)
(c)

Used effects		Solved contradictions
1.1	Fixed springs	Productivity-Reliability
2.1	Linear moving	Productivity-Force
3.3	Extendable sub plate	Productivity-Reliability
4.3	Foldable handle	Productivity-Weight of st. Productivity-Device complexity
5.1	Rack and pinion	Productivity-Reliability Productivity-Device complexity
6.1	Inclined form	Productivity-Length of st.

Fig. 22: CV 2 : (a) geometry representation ; (b) used effects ; (c) solved contradictions (cf. [18,45]).

Based on Table 16, the SPs applied in CV 1 and CV 2 are resumed in Table 17 with respect to the used effects (cf. Fig. 21b and Fig. 22b). As done in Fig. 21 and Fig. 22, the differing effects and the SPs are highlighted in red.

Table 17: SPs applied in CV 1 and CV 2 with respect to the used effects.

CV 1		CV 2	
Used effect	**Applied SP**	**Used effect**	**Applied SP**
1.1 Fixed springs	Prior action (10)	**1.1** Fixed springs	Prior action (10)
2.1 Linear moving	Dynamicity (15)	**2.1** Linear moving	Dynamicity (15)
3.1 Angular sub plate	Prior action (10)	**3.3** Extendable sub plate	Prior action (10)
4.1 Extendable handle 1	Nesting (7) + Local quality (3)	**4.3** Foldable handle	Segmentation (1) + Local quality (3)
5.1 Rack and pinion	Segmentation (1)	**5.1** Rack and pinion	Segmentation (1)
6.1 Inclined form	Spheroidality (14)	**6.1** Inclined form	Spheroidality (14)

By comparing Fig. 21c and Fig. 22c with Table 15, we can see that all the contradictions identified for the current system are solved at least one time for both CV 1 and CV 2. This means all the costumers' needs (i.e. the requirements, cf. Fig. 19b) have been fully satisfied. As highlighted in Fig. 21 and Fig. 22, the two generated CVs differ in the effects used to realize the subfunctions "3. Secure the system" and "4. Transmit human force". Notably, for the sub-function 3, the use of an extendable sub plate (CV 2, cf. Fig. 22b) instead of an angular sub plate (CV 1, cf. Fig. 21b) does not affect the corresponding solved contradiction (i.e. productivity-reliability, cf. Fig. 21c and Fig. 22c). On the other hand, for the sub-function 4, using a foldable handle (CV 2, cf. Fig. 22b) instead of an extendable handle (CV 1, cf. Fig. 21b) means applying a different SP (segmentation rather than nesting, cf. Table 17), and this consequently change the solved contradiction (cf. Fig. 21c and Fig. 22c).

1.3.2.3. The Principles of the Adopted Approach (PAAs)

Based on TIPS (cf. sections 1.3.2.1 and 1.3.2.2), the following points constitute the PAAs which will characterise the development and the methodology of the demonstrator presented in this thesis:

- Model of conceptual design procedure. According to Fig. 16 and Fig. 17, our aided design procedure will be composed of two main steps: the identification of the solutions principles (SPs) and the development of the concept variants (CVs). As anticipated, a minor importance will be given to the final evaluation of the CVs, since our main purpose is to be able to generate as many different CVs as possible (cf. Table 9).
- Use of equation-free design methods. As shown, the conceptual design methods proposed in TIPS does not employ scientific equations (behaviour models, as Eqs. (1), (2), (5) and (6)). Instead design knowledge are made available by means of SPs, effects and design cases and solutions are thus provided as CVs (cf. Fig. 21 and Fig. 22), in a non-numerical form. Based on Table 11, we believe that including similar methods (i.e. which do not employ scientific equations) in our approach can involve a minor focus on technical aspects and on optimal solutions, making unconventional solutions more likely, as suggested in the BDPs (cf. Table 5). This will also reduce the influence of designer's cognitive biases, since all the choices linked to equations and design parameters are no more required, in accordance with our purposes (cf. Table 9).
- Use of the natural language (dialectics) to identify and express the SPs and the CVs. In TIPS, the SPs are based on recurrent scientific and engineering knowledge and are provided in a dialectic form, i.e. by exploiting the natural language (cf. Fig. 20), as well as the effects (cf. Table 16). As argued in [9,46], dialectics is immediate, simply accessible and it emphasizes the change (i.e. innovation, cf. Fig. 7), and it can thus easily be used to eliminate the contradictions among the required specifications. Moreover, the natural language provides the possibility to admit multiple and very different solutions which can significantly enlarge the solution space, as indicated in the introduced BDPs (cf. Table 5). Based on Table 9, all these aspects are in accordance with our

purposes. In a similar way, the methods used in our demonstrator to identify the SPs and the CVs will be based on the manipulation of the natural language.

- Use of illustrated design examples to support the application of the SPs and the development of the CVs. As seen for TIPS, the visual representations of the design cases provided with the morphological matrix (cf. Table 16) constitute a powerful mean to quickly transmit multiple and diverse knowledge to the designer [8,11,43]. Moreover, according to Table 16, same SPs and effects can be applied in different ways, thus based on different design cases. This aspect contributes to further enlarge the solution space, in accordance with the BDPs, and thus with our purposes (cf. Table 9). In a similar way, the implemented demonstrator will provide a catalogue of illustrated design examples linked to the SPs, to help designer generating different CVs.

The introduced PAAs and their corresponding fundamental aspects are reported in Table 18. All points in Table 18 have a role in favouring the BDPs and contribute to achieve our purposes.

Table 18: Principles of the adopted approach (PAAs) and corresponding fundamental aspects.

PAA	Fundamental aspects
Model of conceptual design procedure (cf. Fig. 16): Step 1: Identify solution principles (SPs) Step 2: Develop concept variants (CVs)	Minor importance of CVs' evaluation ➔ Keep also less promising solutions (BDPs)
Use of equation-free design methods • No scientific equations employed • Solutions provided in a non-numerical form	Minor focus on technical aspects and optimal solutions ➔ unconventional solutions are more likely (BDPs) Reduce influence of the cognitive biases
Use of the natural language to express and identify the SPs and the CVs	Immediacy and emphasis on change (i.e. innovation, cf. Fig. 7) More different solutions (BDPs)
Use of illustrated design examples (visual representations of different design cases) to support the application of the SPs and the development of the CVs	Quick transmission of multiple and different knowledge to the designer More different solutions (BDPs)

2. Machine learning methodology

In the previous chapter, we presented the purposes pursued in the present work (cf. Table 9) and the principles of the adopted approach (PAAs, cf. Table 18). This chapter is devoted to the developed machine learning methodology. Machine learning represents an important branch of the computational algorithms' domain, and it involves different techniques to reproduce human intelligence by emulating different cognitive abilities [47]. Before illustrating the implemented machine learning methods which will be integrated in our demonstrator, it is crucial to us to specify why we adopted machine learning in this work. This enables the reader to better understand our standpoint. The conceptual map of the chapter is reported in Fig. 23 and highlights the fundamental topics on which the discussion is conducted. As shown in Fig. 23, the chapter is organized in two main sections:

- A preliminary introduction (section 2.1), in which we study the role of the key cognitive abilities (KCAs) in conceptual design, and where we present the advantages offered by using machine learning by comparing brain and machine.
- The technical description of the developed machine learning methods (section 2.2).

In section 2.1 ("preliminary introduction", cf. Fig. 23), a small overview on brain basic biological aspects, regarding neurons and synapse, and on memory is firstly proposed. The objective is to understand how information is stored in our brain and retrieved to awareness, also introducing many important points which will be recalled in the following sections. Based on the adopted conceptual design procedure (cf. Fig. 16 and Table 18), we then focus on the KCAs, i.e. the cognitive abilities responsible for our great performances of learning and knowledge manipulation. An original representation of the designer mental process is constructed to explain the crucial role of the KCAs during conceptual design, in accordance with the thesis purposes and the PAAs (cf. Table 9 and Table 18, respectively). It emerges that KCAs are potentially very suitable to favour the development of breakthrough design solutions (cf. Fig. 23). We finally propose an original comparison between brain and machine, highlighting the limits of the brain to conceptual design, including cognitive biases (cf. Table 6 and Table 7), which however do not affect machines (cf. Fig. 23). The base idea linked to the use of machine learning in our methodology is finally pointed out at the end of section 2.1, and it is also reported in Fig. 23: exploiting the power of KCAs without being affected by brain limits. Due to this ambitious idea, the machine learning methods developed in this work will mainly focus on the identification of the SPs, i.e. on the first step of the adopted conceptual design procedure (cf. Table 18).

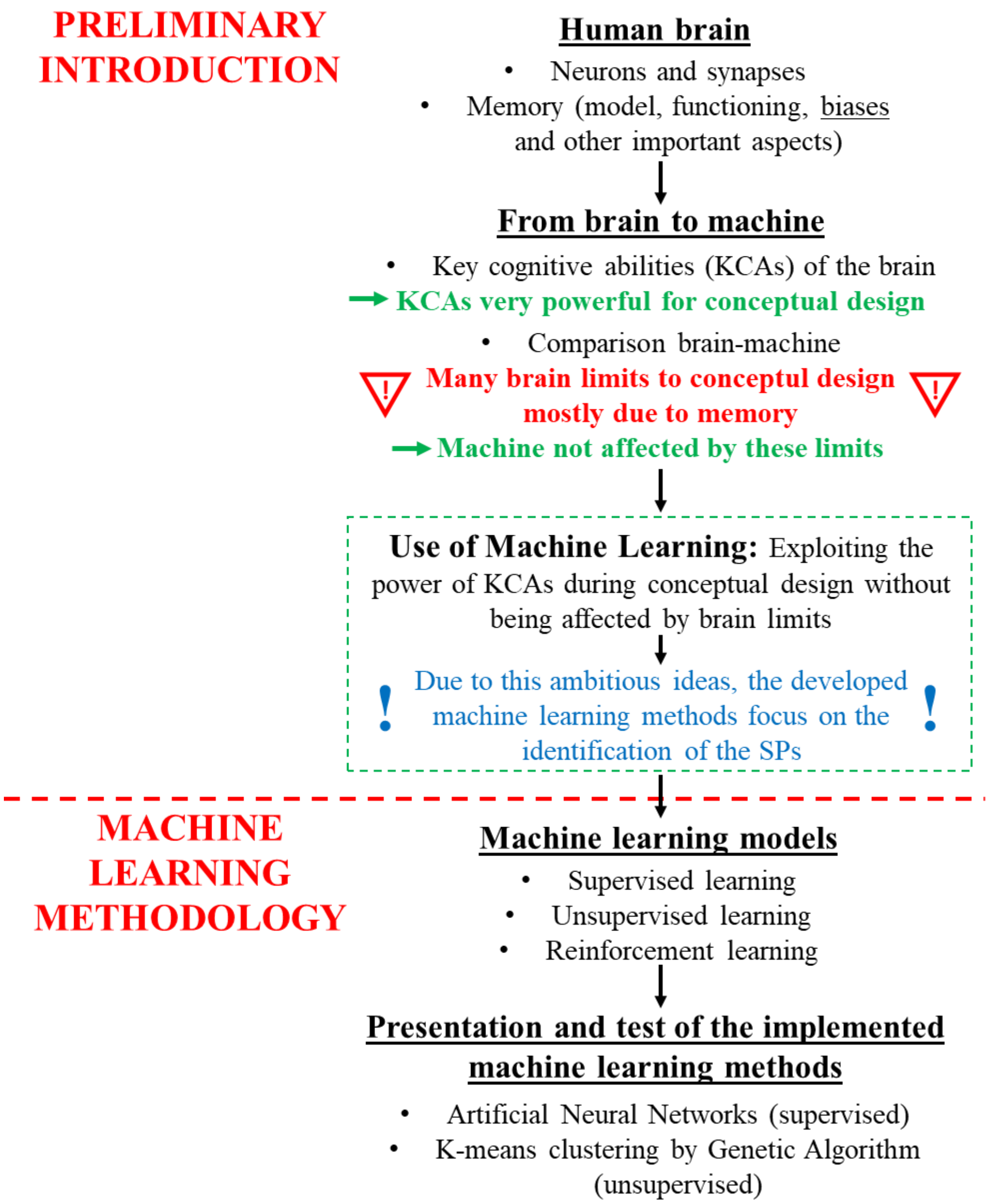

PRELIMINARY INTRODUCTION
Human brain
Neurons and synapses
Memory (model, functioning, biases and other important aspects)
From brain to machine
Key cognitive abilities (KCAs) of the brain
KCAs very powerful for conceptual design
Comparison brain-machine
Many brain limits to conceptual design mostly due to memory
Machine not affected by these limits
Use of Machine Learning: Exploiting the power of KCAs during conceptual design without being affected by brain limits
Due to this ambitious ideas, the developed machine learning methods focus on the identification of the SPs
MACHINE LEARNING METHODOLOGY
Machine learning models
Supervised learning
Unsupervised learning
Reinforcement learning
Presentation and test of the implemented machine learning methods
Artificial Neural Networks (supervised)
K-means clustering by Genetic Algorithm (unsupervised)

Fig. 23: Conceptual map of the chapter developed to explain the original path leading to the implemented machine learning methods.

In section 2.2 ("machine learning methodology", cf. Fig. 23), the three main machine learning models[5] are immediately introduced. Based on two of these models, two different machine learning methods were implemented in MATLAB [48], i.e. artificial neural networks (ANNs) and K-means clustering, which are thus presented following the order in Fig. 23. Both methods are first theoretically described, outlining their base mechanisms and highlighting the influence of the main parameters, and they are finally tested and validated with original examples.

[5] The learning model identifies the human cognitive ability (or the aspect of the human intelligence, in general) to be emulated. The terminology related to machine learning is lately discussed in section 2.2.1.

2.1. The brain: an extraordinary but "imperfect" machine

In the current section, we present the base ideas of our methodology, also introducing the reasons for adopting machine learning. An overview on brain and memory is firstly provided in section 2.1.1, to understand the basic principles of learning and retrieving (i.e. how learnt information is recalled to awareness). The powerful cognitive abilities of the brain are then presented and discussed in section 2.1.2. In the same section, a brain-machine comparison is finally proposed to highlight brain limits regarding our purposes (cf. Table 9) and the advantages offered by machines. The latter comparison represents a crucial point of our work.

2.1.1. An overview on brain and on memory

The brain is a dense network constituted by billions of elementary processing units (or cells), called neurons, connected by means of synapses. As reported in [49,50], our brain is characterised by approximately 10^{11} neurons, each one forming 500-1000 synapses. Based on these numbers, the total amount of synapses in the brain can be estimated between 10^{13} and 10^{15}. This enormous connectivity potential is the source of our extraordinary cognitive and learning capabilities.

2.1.1.1. The neuron: the elementary processing cell of the brain

A neuron can be considered as a sort of switch, which can activate (or not) as function of the information received. The neuron anatomy is illustrated in Fig. 24a, adapted from [49,50]. According to Fig. 24a, the different input signals from other neurons are collected by the dendrites and accumulated into the nucleus as an electrical potential. Input signals can be stimulating or inhibiting, and if the accumulated potential exceeds a specific threshold value the cell nucleus activates by emitting an electrical impulse. The impulse pass through the axon and is transmitted to other neurons, or cells with different functions, by means of synapses at the axon terminals (cf. Fig. 24a). The axon is electrically isolated, and it can achieve lengths of 1 m, enabling the neuron to communicate with other systems and organs of the body, as the muscles [49,50].

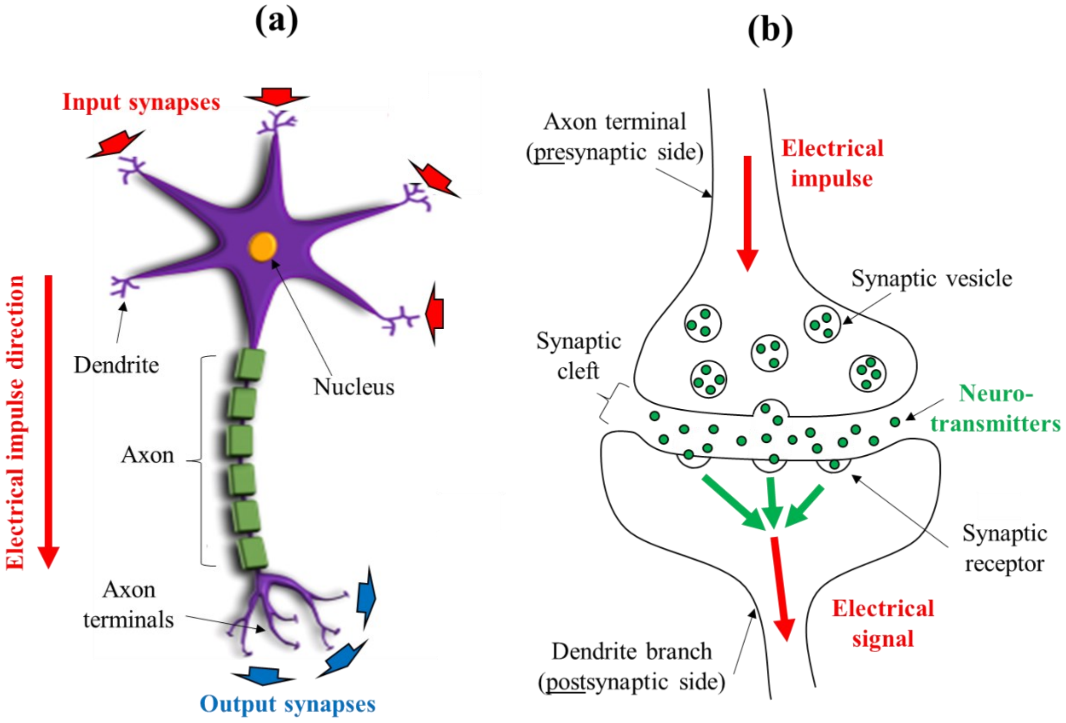


Fig. 24: (a) anatomy of a neuron and (b) representation of a chemical synapse [49–51].

2.1.1.2. The synapse: the special neuronal connection

The synapse is a special connection allowing neurons to communicate each other or with other different cells, as muscular or sensorial cells. Two main variants of synapse can be distinguished: electrical and chemical synapse. In both cases, the transmitting neuron is called presynaptic neuron, while the receiving cell is called postsynaptic cell.

- The electrical synapse enables a direct and quick communication by means of electrical signals, which are transmitted through the presynaptic axon terminals to the postsynaptic cell nucleus. The electrical synapse occurs when a rapid and synchronic response by a great number of neurons is required, as for example in danger situations or in case of quick reflexes [50].
- The chemical synapse is the more distinctive in humans, and it is represented in Fig. 24b according to [50,51]. As illustrated, the electrical impulse transmitted by the presynaptic neuron reaches an axon terminal. Here, it is converted into a "chemical signal", by inducing the release of the neurotransmitters, specific molecules incapsulated in the synaptic vesicles. The neurotransmitters cross the synaptic cleft towards the postsynaptic dendrite, where they are absorbed by the synaptic receptors and converted into an electrical signal for the nucleus (cf. Fig. 24b). The great advantage of the chemical synapse resides in the "adjustability" of the connection: a large variety of neurotransmitters exists, and they can be released in different quantities, allowing for stronger or weaker connections (i.e. "weighted" connections) [50,51].

The adjustability of the chemical synapses is the base of the brain learning ability. The principal aspects of the analysed synaptic variants are resumed in Table 19.

Table 19: The two synaptic variants and their principal aspects.

Synapse variant	Description	Principal aspects
Electrical synapse	Direct and quick communication by means of electrical signals	It allows for a rapid and synchronic response by a great number of cells (reflexes)
Chemical synapse	Release of neurotransmitters across the synaptic cleft, stimulated by the presynaptic electrical impulse (cf. Fig. 24b)	Adjustable ("weighted") connection: variable nature and quantities of the released neurotransmitters (this is the base of the learning process!)

2.1.1.3. Introduction to memory: our personal store of knowledge

As reported in Table 19, the adjustability of the chemical synapses is the base of the learning process, which involves the acquisition of knowledge and their synthesis into encoded information. The ensemble of all the encoded information constitutes the memory, i.e. our personal archive of past experiences. The most popular model of the memory was proposed in [52] by Atkinson and Shiffrin, in 1968, and it is represented in Fig. 25. Although many improvements in this field have been achieved in more recent works, this model remains in regular use nowadays, since it is easily comprehensible and widely supported by experimental evidence in literature [53–56]. According to this model, the memory is organized in a multi-store structure, consisting of three separate levels (or stores, cf. Fig. 25): sensory memory (SM), short-term memory (STM) and long-term memory (LTM). Notably, Fig. 25 describes how information is passed and processed through these three memory stores, each one having different characteristics in terms of:

- Duration of the encoded information.
- Store capacity, concerning how much information can be stored.
- Type of encoded information (visual, acoustic, semantic, etc.).

The characteristics of each store have been reported in Table 20, according to [52,56].

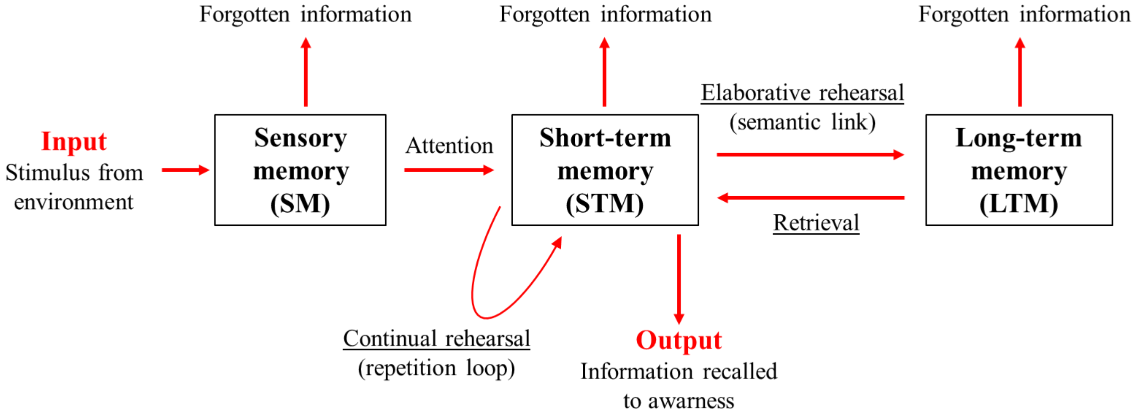


Fig. 25: Multi-store memory model of Atkinson and Shiffrin [52,56].

Table 20: Characteristics of the memory stores in the model of Atkinson and Shiffrin [52,56].

Memory store	Duration of encoded information	Store capacity	Type of encoded information
Sensory memory (SM)	0.25 – 0.5 seconds	Large (all sensory experience)	Visual (sight), auditory (sounds), touch, taste, smell
Short-term memory (STM)	● 0 – 18 seconds (normally) ● Up to 30 seconds, with continual rehearsal (cf. Fig. 25).	3-5 chunks[6]	Mainly auditory
Long-term memory (LTM)	Few minutes – lifetime (difficult to define)	Theoretically unlimited	Explicit ● Mainly Semantic ● Visual (images) ● Auditory Implicit ● Skills and habits ● Emotions, reflexes ● Objects' and language models (cf. Fig. 26, section 2.1.1.6)

As shown in Fig. 25, each stimulus we perceive through the five senses is initially stored in the SM. Notably, information derived from an input stimulus activates an enormous number of synapses (synaptic adjustment, cf. Table 20), forming a sort of "neuronal path" which constitutes the trace of the information encoded in the SM. Most of the perceived stimuli receives no attention, and it is thus retained for a very brief period (cf. Table 20). They are thus forgotten (cf. Fig. 25) by "overwriting" the

[6] Based on [57,58], the word "chunk" identifies a group of information pieces connected by a common pattern. For instance, let us imagine memorizing the number 9876534: the sequence of 9-8-7-6-5-3-4 (composed of seven distinct information elements) would be chunked, for example, into 987-6534 (i.e. into two larger blocks of information). The natural chunking ability of our brain enables to compress large information in few blocks, thus storing more information at once in the STM.

corresponding neuronal path. As reported in Table 20, the SM has a large capacity, and it can encode information from all the five senses. Based on [52,53], the passage of information through STM and LTM is described in sections 2.1.1.4 and 2.1.1.5. The processes of rehearsal and retrieval (cf. Fig. 25) are summarised in Table 21. A deeper analysis of the LTM, regarding the type of encoded information (explicit and implicit, cf. Table 20), is provided in section 2.1.1.6.

2.1.1.4. The passage through the STM and the process of continual rehearsal

Focusing the attention induces the transfer of information to the STM (cf. Fig. 25). Notably, the STM constitutes a temporary working space where information incoming from the SM is elaborated and prepared for different tasks [59]. As reported in Table 20, the store capacity of the STM is limited to 3-5 chunks[6] (cf. [57,58]), and the duration of the encoded information does not normally overcome the 18 seconds. This duration can be however extended up to 30 seconds by means of the continual (or maintenance) rehearsal, i.e. the process of verbally (or mentally) repeating information (cf. Fig. 25). Notably, the information is repeated without thinking about its meaning or connecting it to other information [52]. This explains why the type of information encoded in the STM is mainly auditory (cf. Table 20) [54]. This type of rehearsal involves the continual adjustment and “regeneration” of the synaptic trace (cf. Table 19) corresponding to the repeated information, making it stronger. We typically perform continual rehearsal when we memorize information to be used for a current purpose, as a telephone number to be called in the immediate future, for example. If repetition does not occur, we will quickly forget it (i.e. information expires from the STM). The principal aspects of continual rehearsal are summarised in Table 21. As shown in Fig. 25, the STM continually communicate with the LTM through elaborative rehearsal and retrieval. This latter aspect is presented in section 2.1.1.5.

2.1.1.5. The passage through the LTM: elaborative rehearsal and retrieval

Giving meaning to information favours its transfer in the LTM, i.e. the archive containing our knowledge, with a theoretically unlimited capacity (cf. Table 20). Let us suppose, for example, that the above phone number (cf. section 2.1.1.4) belongs to our dear friend, who we well know (i.e. whose information is thus already encoded in our LTM). Not only we often call this number, but we also associate a meaning to it, i.e. “belonging to our dear friend”. This process ensures that information is encoded in the LTM and is called elaborative rehearsal (cf. Fig. 25). Moreover, some information about our dear friend will be immediately recalled to awareness, as his image or the date of his birthday: this is a consequence of the retrieval process (cf. Fig. 25). According to Fig. 25, elaborative rehearsal and retrieval form a continual information cycle between the STM and the LTM. Notably, not only the information incoming from the STM can be encoded in the LTM, but it can also potentially decode (i.e. retrieve) some information already encoded in the LTM [52,60]. The bidirectional interaction between STM and LTM can be better understood by comparing the STM to a computer desktop, and the LTM to the hard disk (i.e. the storage unit). As a desktop, the STM represents a working space, where information is temporarily manipulated and prepared to execute several tasks [59] (cf. section 2.1.1.4). As a hard disk, the LTM constitutes our stable archive of information, which is continually used and modified during all the performed tasks. Indeed, we do not normally see on the desktop all the files stored in the hard disk, as well as we do not normally see the overall information available in the LTM. Let us provide more details about elaborative rehearsal and retrieval.

- **Elaborative rehearsal**. This process consists of linking new information, incoming from the STM, with information already stored in the LTM in a meaningful way [52]. Notably, elaborative rehearsal is more effective than continual rehearsal, since it involves the construction of new synapses, which ensure a much more stable information encoding than STM. The stability of encoded information over time also depends on the motivational and emotional state of the subject. This makes difficult to define the storing duration, which can vary from few minutes to a lifetime (cf. Table 20) [60].

- **Retrieval**. Information decoding (retrieving) requires a similitude (associative) relationship between the information incoming from the STM and the information to be decoded. The incoming information is "compared" to the synaptic trace of encoded information: the more similitudes they share, the higher the probability the latter is decoded and recalled to the STM (i.e. "visualised" on our desktop) [61]. This physically occurs with the reactivation of the synaptic traces (neuronal path) corresponding to the decoded information [62]. As seen for elaborative rehearsal, the retrieving of encoded memories also depends on motivational and emotional state of the subject.

The processes of rehearsal and retrieval are summarised in Table 21.

Table 21: The processes of rehearsal and retrieval (cf. Fig. 25) and their principal aspects.

Process	Description	Associated synaptic mechanism
Continual (or maintenance) rehearsal (cf. section 2.1.1.4)	Verbally (or mentally) repeating information (repetition loop, cf. Fig. 25), in general without thinking about its meaning	Continual adjustment ("regeneration") of the synaptic trace corresponding to the repeated information
Elaborative rehearsal	Linking new information with information already encoded in the LTM by means of meaning. Influenced by the emotional and motivational state.	Construction of new synapses which ensure a stable information encoding
Retrieval	Information incoming from the STM is "compared" to information encoded in the LTM: the more similitudes they share the higher the probability the encoded information is decoded and recalled. Influenced by the emotional and motivational state.	Reactivation of the synaptic traces corresponding to the encoded information

2.1.1.6. LTM: explicit and implicit knowledge

In the current section, we provide more details about the type of information encoded in the LTM (cf. Table 20), to better understand what kind of knowledge we can store and retrieve. Since the 1970s, many models of the LTM have been developed, typically by defining different LTM sub-categories, with different roles and characteristics [63–65]. The model represented in Fig. 26, proposed in [63], was one of the first and most influential ones. As shown in Fig. 26, the LTM can be divided into two parts: an explicit knowledge, also called declarative (or conscious), and an implicit knowledge, also denoted as non-declarative (or unconscious). Both parts are in turn divided into two sub-categories. Although more recent models propose a greater number of sub-categories, they always retain the same distinction between explicit and implicit knowledge [64–66].

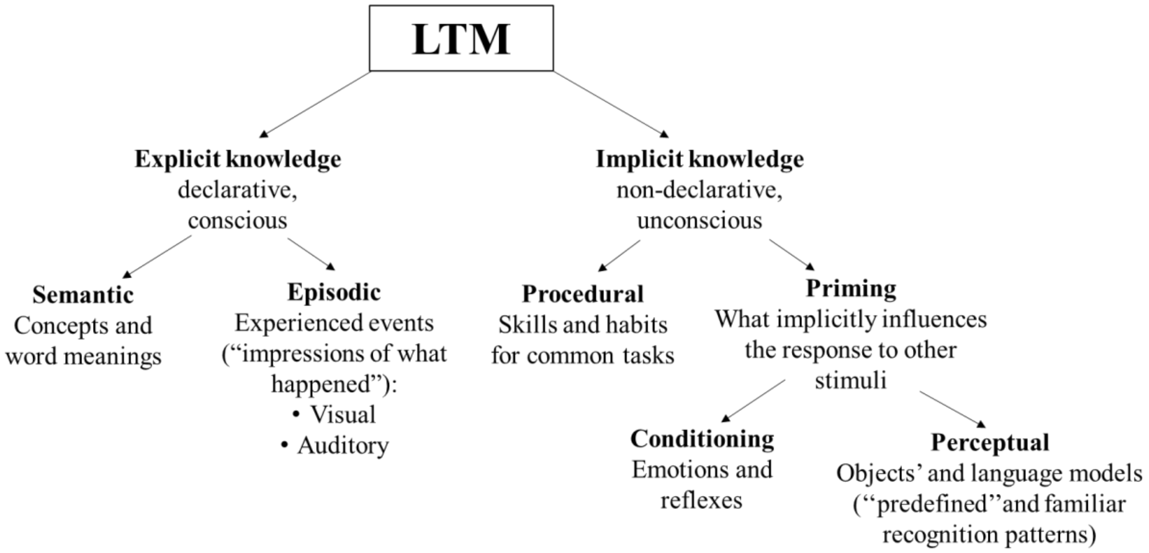


Fig. 26: Influential model of the LTM [63].

According to [63], the explicit knowledge involves conscious effort to encode and recall information, and it is mainly categorised into (cf. Fig. 26):

- Semantic memory, which includes the knowledge about concepts and word meanings, as well as general knowledge. For instance, we know that Rome is the capital of Italy because we "hold" a semantic link (elaborative rehearsal, cf. Fig. 25 and Table 21) between "Rome" and "to be capital of Italy", and this information is consciously "declarable". As reported in Table 20, the semantic memory constitutes the main part of the explicit knowledge.
- Episodic memory, which contains information about experienced events, regarding for example: specific moments of our life (called autobiographical memories, as our first day of school), general feelings associated to an activity (for example, the feeling associated to running), and exceptionally vivid and detailed "snapshots" linked to important or surprising circumstances (called flashbulb memories). All this knowledge is consciously declarable, and it is mainly encoded as visual and auditory information (cf. Table 20).

On the other hand, the implicit knowledge involves unconscious and automatic (and thus not consciously controlled) thoughts, and it includes all we have implicitly or "accidentally" stored. The implicit knowledge is typically distinguished into (cf. Fig. 26):

- Procedural memory, including skills and habitudes linked to the tasks we commonly execute without conscious efforts. For instance, the memory of the motor skills for riding a bike, or the habitude to brush the teeth every morning.
- Priming memory, including all the factors which unconsciously (i.e. implicitly) influence the response to other stimuli. According to [63–66], two main components of the priming memory can be distinguished (cf. Fig. 26). The first one, called conditioning memory, determines all the emotional and reflex based behaviours. The second one, called perceptual memory, involves objects and language models (or structures, cf. Table 20).

Notably, the models stored in the perceptual memory (cf. Fig. 26) constitute "predefined" and familiar recognition patterns which allow to speed up the learning process.

2.1.1.7. Final considerations: important aspects concerning memory and its functioning

We have discussed how information is processed by the brain, analysing the main mechanisms enabling new information to be encoded in the memory and which allow stored knowledge to be retrieved and made available. Based on the presented elements, we want to finally highlight some important aspects regarding memory and its functioning, in order to better introduce machine learning in the next sections.

- **Continual modification and re-elaboration of the stored knowledge**. As seen in section 2.1.1.5, the retrieval process (cf. Fig. 25) involves the decoding of the information stored in the LTM (our hard disk), by reactivating their synaptic traces, and their recalling to the STM (our desktop). Due to the continual information cycle between STM and LTM the same information just retrieved can be redirected towards the LTM, thus re-processed and re-encoded by elaborative rehearsal. This phenomenon is called reconsolidation: retrieved memories are actively modified and re-elaborated, and thus re-encoded by means of new synaptic traces [67]. Due to reconsolidation, some retrieved memories can be completely rewritten, or even erased. The synaptic mechanisms linked to this phenomenon are still unclear [68]. However, this supports the idea that our memory should not be considered as a passive archive containing static information, but rather as an active archive whose stored knowledge is under continual modification and re-elaboration [69].
- **Loss of the capability to access (retrieve) some stored information over time**. As reported in Table 20, the capacity of the LTM is thought to be theoretically unlimited, which means an infinite amount of information could be potentially encoded, and thus be available by retrieval (cf. Fig. 25) [52,53]. Rather than availability, the main constraint regards the accessibility to the stored knowledge. According to Table 21, the access to the stored knowledge reposes on the similarity between the information to be decoded and the retrieval information incoming from the STM. The retrieval signal can be consciously elaborated in order to recall specific encoded information. However, the capability to retrieve some memories can be gradually lost over time, due to the loss of the neuronal paths which allow to access them. As argued in [70], this mechanism occurs unintentionally (unconsciously), and it explains why older memories are generally harder to retrieve than more recent ones. Notably, this does not necessarily mean that those memories have been removed: they are probably still available in the LTM, but we are no more able to access them.
- **Implicit (unconscious) effects of emotional conditioning on learning and retrieving**. Based on Fig. 26, the responses to received stimuli are generally influenced by the priming memory, on which we have a limited cognitive control due to its non-declarative nature [63]. The effect of the emotional conditioning component (cf. Fig. 26) on the elaborative rehearsal and retrieval processes has been already highlighted in Table 21. These effects can be both positive and negative. For example, it is widely demonstrated that certain emotions can facilitate learning and memorization. A similar effect can be observed for retrieving: the emotional content of some perceived stimuli can potentially facilitate the retrieval of specific and detailed memories. Clearly, these abilities differ from a person to another, since emotions are extremely personal. On the other hand, specific emotions can also be responsible of unwanted memory suppression (psychological trauma, for example), whose mechanism is however still unclear [71].
- **Conscious and unconscious (or cognitive) memory biases**. According to [72–74], two types of memory biased attitudes (or biases) can be distinguished: unconscious (or cognitive) biases (cf. Table 6), due implicit knowledge, and conscious biases, due to explicit knowledge (cf. Fig. 26). As widely discussed in the previous (cf. section 1.2.3), the cognitive biases operate outside our awareness, and involve negative effects on our behaviour and design habitudes (BDHs, cf. Table 7 and Fig. 11). Notably, three principal cognitive biases have been identified in Table 6: status quo,

functional fixedness, and confirmation. Based on literature, it is not easy to establish whether each one of these biases belongs to procedural rather than priming memory. They should be probably considered as spread all over the implicit memory. Nonetheless, we believe some connections could be found, for example, between confirmation and functional fixedness biases and the perceptual memory (cf. Fig. 26). As reported in Fig. 26, the perceptual memory is characterised by models of objects and language, which play a fundamental role in speeding up the learning process. On the other hand, these models constitute familiar patterns (functional fixedness, cf. Table 6) which unconsciously involve a selective learning: information "fitting" with these models (confirmation bias, cf. Table 6) is permanently retained in the LTM, while the rest risks to be early forgotten [71]. Contrarily to cognitive biases, conscious biases are explicitly and semantically processed, mostly as words and concrete actions. Consequently, conscious biases manifest in a more overt way, as for example physical and verbal harassment or different modalities of social exclusion (sexism, racism and other types of discriminations) [73,74]. Since conscious biases are explicitly expressed, we are thus more aware of them than cognitive biases, which are instead typically unknown and unnoticed. Regarding our purposes in this thesis (cf. Table 9), we believe cognitive biases are much more dangerous than conscious due to their implicit nature. Nonetheless, conscious biases are important as well to understand how memory operates.

The discussed aspects are summarised in Table 22.

Table 22: Important aspects regarding memory and its functioning.

Memory aspect	Description
Continual modification and re-elaboration of the stored memories	Reconsolidation: retrieved memories are modified and re-written, i.e. re-encoded in a different way. The LTM is an active archive: it does not contain static, but rather continually modified and re-elaborated information.
Loss of the capability to access (retrieve) some encoded information over time	Loss over time of the neuronal paths which allow to access some information. This information remains potentially available, but it is no more retrievable.
Implicit (unconscious) effects of emotional conditioning on learning and retrieving	Positive effect: facilitated learning and retrieving for information which holds a certain emotional content. Negative effect: unwanted memory suppression due to specific emotions.
Conscious and unconscious (or cognitive) memory biases	Unconscious (or cognitive) biases (cf. Table 6, Table 7 and Fig. 11): mental mechanisms negatively affecting our behaviour and operating outside our awareness (implicit knowledge). • Confirmation and functional fixedness biases: selective learning due to the objects and language models which characterise the perceptual memory (cf. Fig. 26). Conscious biases: negative attitudes explicitly and semantically processed as words and concrete actions. • Examples: physical and verbal harassment, different types of social exclusion (sexism, racism, etc.). Regarding our purposes (cf. Table 9), cognitive biases are more dangerous than conscious biases due to their implicit nature.

Since each individual has different memory performances, the weight of these aspects can be different from one person to another [71]. According to Table 22, we must however keep in mind that there exist many factors in our memory we cannot explicitly control, and which represent a potential obstacle to the achievement of our purposes.

2.1.2. From brain to machine

The analysis of memory conducted in section 2.1.1 allowed to better understand how knowledge is stored in our LTM and made available to our brain. According to the proposed memory model (cf. Fig. 25), input information is processed through three different memory stores (cf. Table 20). Thanks to the bidirectional communication between STM and LTM (cf. Fig. 25), input stimuli can be encoded as permanent knowledge (elaborative rehearsal, cf. Table 21), and contemporarily enable encoded information to be retrieved as output (retrieval, cf. Table 21). Regarding conceptual design (cf. Fig. 16), the input stimulus is constituted by initial idea and specifications, while the retrieved (and elaborated) output is represented by the developed CVs. Based on the example illustrated in Fig. 27, let us assume to cognitively perform a conceptual design procedure by exploiting the design knowledge stored in our brain. In this case, the input stimulus involves a cantilever beam with a full squared section (initial idea), and the reduction of its mass (specification). These elements are thus processed by our brain, which quickly proposes two different CVs.

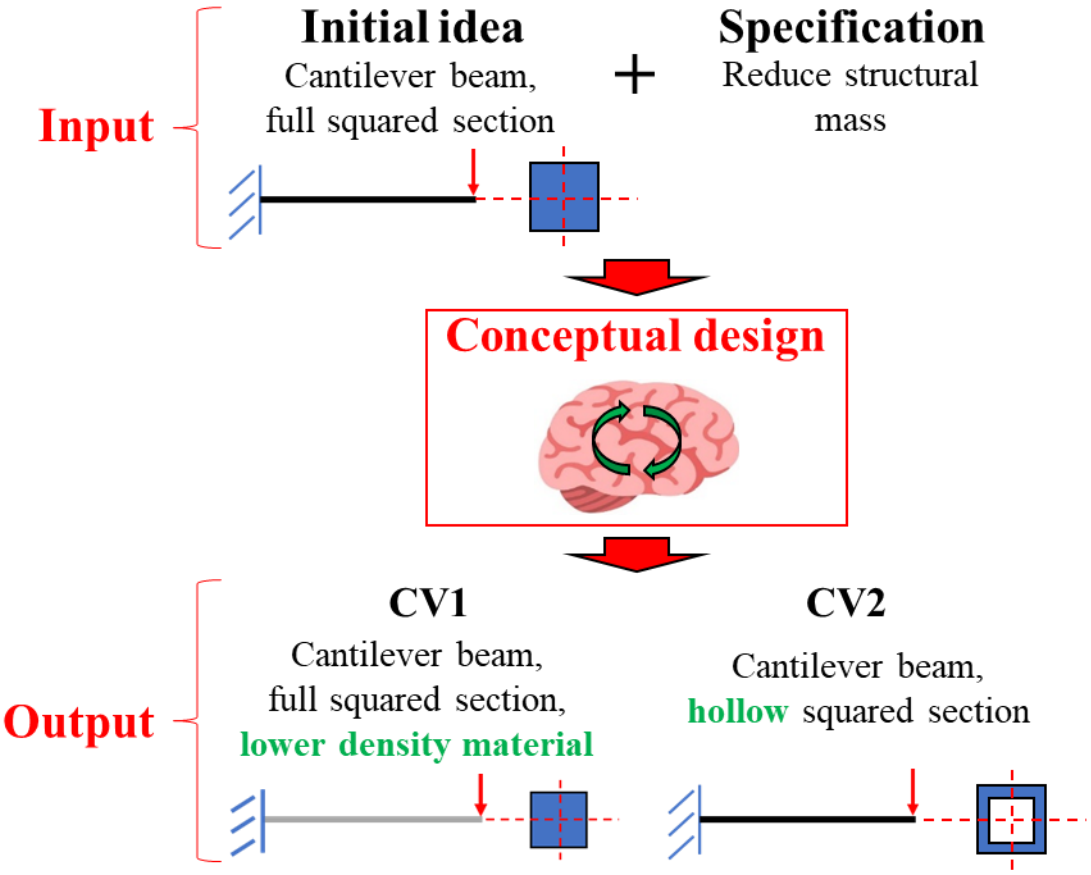

Input
Initial idea
Cantilever beam, full squared section
+
Specification
Reduce structural mass
Conceptual design
Output
CV1
Cantilever beam, full squared section, lower density material
CV2
Cantilever beam, hollow squared section

Fig. 27: Example of conceptual design performed by the brain.

As shown in Fig. 27, both inputs and outputs are characterised by:

- A dialectic description (use of natural language).
- A visual representation of the design configuration.

These characteristics involve the main categories of the explicit knowledge (cf. Fig. 26), i.e. the semantic memory and the episodic memory, respectively. Moreover, they immediately emphasise the changes of the CVs from to the initial idea, in accordance with the PAAs reported in Table 18. The green bolded words (cf. Fig. 27) highlight the SPs on which the proposed CVs are based. Regarding CV1, the concept of “lower material density” has been added to the initial idea. In the corresponding representation, the use of a lower density material is indicated by the grey beam. Regarding CV2, its description highlights that the concept of “full”, which characterises the section of the initial idea, has been changed into “hollow”. Both CV1 and CV2 constitute two valid conceptual solutions since they both allow to satisfy the required specification. The following important considerations, based on this example, enable to introduce the organization of the current section.

- The proposed CVs seem two common and immediate solutions. Probably, an experienced brain can achieve them without so much effort, almost automatically. How is that possible? Usually, we do not realize how much powerful our brain is. Notably, some key cognitive abilities, denoted KCAs, can be considered as the main source of our extraordinary brain performances [50,75,76]. In section 2.1.2.1, we want to introduce the KCAs and study their role in conceptual design.
- Although the power of brain performances is unquestionable, the proposed CVs constitute only two possible solutions. We are surely capable to generate other different CVs, based on different knowledge we can retrieve and manipulate. Nonetheless, we will be hopelessly limited by several aspects, mostly due to our own memory (cf. Table 22). Can machine reproduce the extraordinary brain capabilities without being affected by its limits? We will answer this question in section 2.1.2.2, by providing a comparison between the brain and the machine.

Final considerations are given in section 2.1.2.3.

2.1.2.1. The key cognitive abilities (KCAs) and their role in conceptual design

According to [50,75], our performances of learning, reasoning and knowledge manipulation are mainly based on three KCAs: abstraction, generalization and analogy-making. Their generic definitions, adapted from [75–78], are reported in Table 23.

Table 23: KCAs and their generic definitions.

KCA	Generic definition
Abstraction	Ability of mentally isolating a characteristic of a concrete context, to consider the isolated characteristic as a specific analysis object.
Generalization	Ability of associating a similar (or the same) meaning to a variety of similar inputs, regarding for instance objects, experiences, or contexts in general.
Analogy-making	Ability of adapting the knowledge about an already known and familiar context to explain and characterise phenomena occurring in a new unfamiliar context, which however shares one or more similitudes with the first one.

We continually and automatically use these abilities, often without realize it. Their role in conceptual design can be better explained with the help of the example in Fig. 27. Fig. 28 shows an original representation of the mental processes which led to CV1 and CV2 (cf. Fig. 27), highlighting the intervention of the KCAs in the two main steps of the adopted conceptual design procedure (cf. Table 18 and Fig. 16). This is one of the possible interpretations of how our brain operates, and it is assumed to explain the roles of the KCAs.

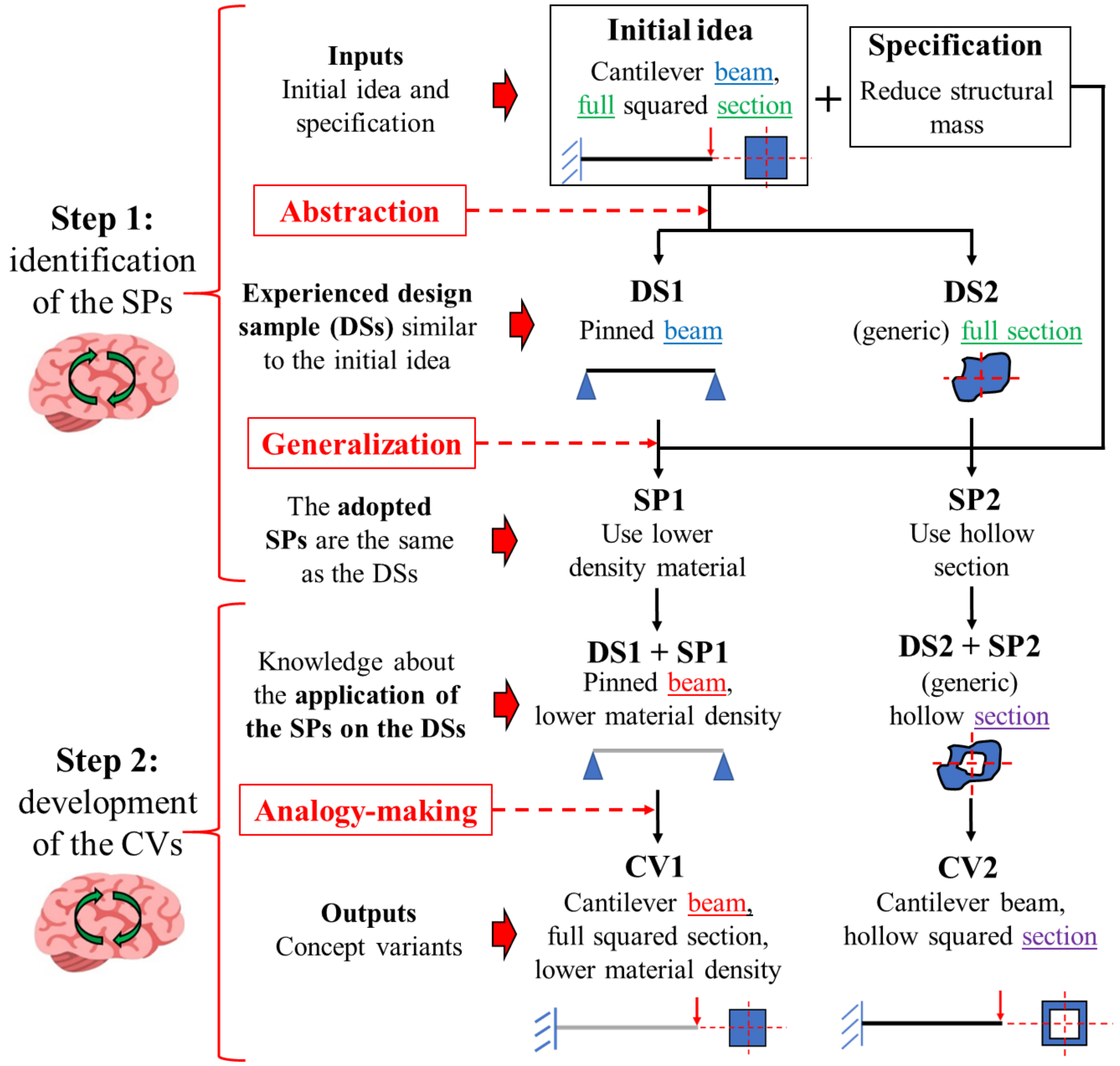

Step 1:
identification
of the SPs
Inputs
Initial idea and specification
Initial idea
Cantilever beam, full squared section
+
Specification
Reduce structural mass
Abstraction
Experienced design sample (DSs) similar to the initial idea
DS1
Pinned beam
DS2
(generic) full section
Generalization
The adopted SPs are the same as the DSs
SP1
Use lower density material
SP2
Use hollow section
Step 2:
development
of the CVs
Knowledge about the application of the SPs on the DSs
DS1 + SP1
Pinned beam, lower material density
DS2 + SP2
(generic) hollow section
Analogy-making
Outputs
Concept variants
CV1
Cantilever beam, full squared section, lower material density
CV2
Cantilever beam, hollow squared section

Fig. 28: Original representation of the mental processes leading to CV1 and CV2, which is assumed to illustrate the roles of the KCAs during conceptual design.

Steps 1 and 2 are thus analysed in detail as follows.

Step 1: identification of the SPs. By abstraction, our brain firstly associates the initial idea to similar experienced design scenarios, denoted DSs (Design Samples, cf. Fig. 28). Notably, the abstraction ability (cf. Table 23) enables to isolate different characteristics of the initial idea, which are thus used to retrieve different DSs by means of similitudes (cf. Table 21). In Fig. 28, the similitudes shared by initial idea and DSs, i.e. "beam" and "full section", have been underlined and highlighted with the same colours (green and blue). In general, the similitudes can involve one or more characteristics of the initial idea. By generalization, our brain then exploits the experiences gained on the DSs to formulate the SPs, as function of the specifications to satisfy. According to Table 23, the generalization ability enables to adopt similar (or the same) responses for similar inputs. In our example (cf. Fig. 28), the same SPs used to reduce the mass of the DSs (which are similar to the initial idea) are adopted to reduce the mass of the initial idea. As also indicated in Fig. 27, the identified SPs involve the use of a lower density material and of a hollow section, instead of full.

Step 2: development of the CVs. As represented in Fig. 28, our brain has already experienced how the identified SPs can be applied to the DSs. As done in Fig. 27, the grey beam (DS1+SP1, cf. Fig. 28) indicates the use of a lower density material. The characteristics of the DSs involved in the application of the SPs, i.e. "beam" and "section", have been underlined and highlighted with two different colours (red and violet, cf. Fig. 28). By analogy based on these characteristics, the knowledge on the application of the SPs is adapted to the initial idea for developing the CVs (analogy-making, cf. Table 23). Notably, the knowledge of applying SP1 to DS1 is adapted to the initial idea based on the "beam" characteristic, thus resulting in CV1 (cf. Fig. 28). CV2 is generated in a similar way: based on the "section" characteristic, the knowledge about the application of SP2 is transferred from DS2 to the initial idea (cf. Fig. 28).

The analysed conceptual design steps are briefly summarised in Table 24 with the involved KCAs.

Table 24: Conceptual design steps and involved KCAs.

Conceptual design step	Description	Involved KCA
1. Identification of the SPs	**1a.** Retrieving of experienced design samples (DSs) similar to the initial idea	Abstraction: The features of the initial idea are isolated and used to retrieve the experienced DSs by similitude
	1b. Exploiting the experiences gained on the DSs to formulate the SPs.	Generalization: the similitudes between initial idea and DSs involve a similar (or the same) response to the required specification
2. Development of the CVs	The knowledge of applying the SPs to the corresponding DSs is adapted to the initial idea for developing the CVs.	Analogy-making: exploiting the analogies (similitudes) between initial idea and DSs for the knowledge adaption.

As assumed in Fig. 28, we did not employ any scientific equation to study the behaviour of the identified DSs, or the influence of the adopted SPs on the initial idea. On the other hand, all the mental operations which led to CV1 and CV2 by exploiting the KCAs were based on similitudes (cf. Table 24), which mostly involved the language used to describe initial idea, DSs and CVs (as the words "beam" and "section", cf. Fig. 28). Indeed, scientific equations contribute to the comprehension of the physical and mechanical phenomena during the construction of our design knowledge. Once the concepts expressed by equations have been learnt and interiorized, the viewpoint adopted in this thesis is that our brain is able to quickly retrieve and manipulate them in a dialectic form and by means of similitudes, without executing any equation. The latter aspects are in accordance with the PAAs (cf. Table 18), which means KCAs are potentially very suitable to favour the development of breakthrough design solutions.

2.1.2.2. Towards machine learning: an original brain-machine comparison

For brevity, only two CVs have been considered in the above example (cf. Fig. 28), as well as only two DSs and only two SPs have been retrieved and manipulated. With a greater conscious effort, we can surely develop novel CVs by means of our KCAs (cf. Table 24). Despite these extraordinary cognitive abilities, the number of novel combinations will be hopelessly limited by several aspects inherent in our brain, as those due to memory (cf. Table 22). Based on these ideas, we need something able to reproduce the KCAs, without however being affected by brain limits. Is machine suitable for this task? With the help of Table 25, let us compare brain and machine. Notably, the comparison terms regard the limits due to memory (cf. Table 22), and two more aspects involving language variety and multidisciplinary abilities, which are directly linked to the PAAs (cf. Table 18).

Table 25: Original brain-machine comparison.

Aspect	Brain	Machine
Conscious and unconscious memory biases	Very dangerous and affecting all the phases of conceptual design Main negative effects: • Prior exclusion of potentially interesting solution paths • Selective learning (cf. Table 22) Very difficult to mitigate (especially unconscious biases)	Potentially introduced in the knowledge base by programmer The more different programmers (specialized in different disciplines) contribute to the knowledge base, the more biases' influence is mitigated
Continual modification of the stored knowledge	Retrieved knowledge is necessarily modified No explicit control on the knowledge base: • Risk of modifying/over-writing the experienced DSs and SPs • Impossible to consciously retrieve some DSs and SPs, since their semantic link could have been modified. • Undesired knowledge cannot be consciously forgotten	No, and the knowledge base can be directly handled: • Copying data before modifying them avoids over-writing • All the available knowledge can be retrieved at will and manipulated ➔ possibility to generate more CVs • Undesired data can be deleted at will
Loss of the capability to retrieve stored knowledge over time	The older the design experiences (DSs and SPs), the less probable they can be retrieved for contributing to conceptual design	No limit in retrieving old data. The knowledge base can be easily transferred to new hard disks, theoretically avoiding the time deterioration of the hardware components
Implicit emotional conditioning	Not consciously controllable and not predictable influence: • Positive effects cannot be easily reproduced • Negative effects cannot be easily avoided	The occurrence of non-predictable and non-controllable (i.e. implicit) events is strongly reduced.
Language variety	Limited: • About 17,000 word families for university graduates • Up to 20,000 word families for really cultured people	Potentially unlimited ➔ A more varied lexicon can involve more different CVs
Multidisciplinary abilities (cf. Table 3)	Often specialized in only one discipline, and rarely in two or more different disciplines • Less analogies between different and apparently unrelated concepts • Difficulty in sharing knowledge with other specialized in different disciplines	Potentially unlimited ➔ analogies between very different and apparently unrelated disciplines are more likely (more different and potentially breakthrough CVs) The more programmers specialized in different disciplines contribute to the knowledge base, the more different analogies are likely

Let us analyse each aspect in detail.

- **Conscious and unconscious (or cognitive) memory biases**. We believe memory biases represent the most dangerous limit of our brain, regarding our purposes (cf. Table 9).

  Brain. According to Table 7, Fig. 11 and Table 22, both conscious and unconscious biases involve the prior exclusion of potentially interesting solution paths due to different reasons. Notably, unconscious biases can also be considered as a potential cause of selective learning (confirmation and functional fixedness, cf. Table 22). Memory biases can heavily affect conceptual design from the earliest phases. For instance, we could be led to priorly fix some characteristics of the initial idea. In general, memory biases are very difficult to mitigate, especially unconscious ones, which are not consciously controllable (cf. Table 22).

  Machine. Theoretically, a machine does not have any biases, even if they could be potentially introduced by the human programmer during the construction of its knowledge base. Nevertheless, the effects of memory biases on thoughts and actions differs from one individual to another [71], and they also depend on the received education (cf. Table 6). We thus believe their presence and influence can be mitigated if more and different programmers, specialized in different disciplines, contribute to the knowledge base of the machine.

- **Continual modification of the stored knowledge**. Due to reconsolidation, retrieving and manipulating design knowledge necessarily involve its continual modification and re-elaboration.

  Brain. Each time DSs and SPs are retrieved and manipulated, they risk being "over-written" (or anyway modified, cf. Table 22). This also implies we do not have a clear and global vision on the overall design knowledge stored in our LTM: we do not know exactly how many and which DSs and SPs we have experienced, since they are under continual modification. Consequently, it could be potentially impossible to consciously retrieve some DSs and SPs: their neuronal path (and thus the corresponding semantic link) could have been changed [67,68]. Moreover, we cannot consciously decide to forget undesired experiences by eliminating their corresponding synaptic trace. In summary, we cannot explicitly control our knowledge base.

  Machine. A machine does not have these problems. Notably, the knowledge base can be directly handled. Data can be copied before being modified (avoiding over-writing) and deleted at will. Moreover, the knowledge base can be completely scanned to consider all the available knowledge: during conceptual design, this enables to enlarge the space of the potential similitudes (cf. Fig. 28), thus increasing the number of generated CVs.

- **Loss of the capability to retrieve stored knowledge over time**.

  Brain. We better remember the DSs and SPs we have recently manipulated, or which we often manipulate. Indeed, recent design experiences can be normally retrieved without so much effort. However, it is more difficult to retrieve older design experiences, since their corresponding neuronal path may have potentially deteriorated over time. Consequently, the older the DSs and SPs are, the less they can contribute to conceptual design: the less novel CVs can thus be developed.

  Machine. Contrary to the brain, a machine has no limit in retrieving old data stored in its knowledge base. Moreover, the knowledge base of a machine can be easily transferred to a new hard disk, avoiding potential damages due to the time deterioration of the hardware components.

- **Implicit emotional conditioning**.

  Brain. Based on Table 22, the implicit and unexpected character of emotions constitute an element of uncertainty and randomness, which can benefit or limit our memory performances in a given context. Their implicit nature makes them not consciously controllable. This means potentially positive effects cannot be easily reproduced, and potentially negative effects cannot be easily avoided.

Machine. A machine has no emotions, which strongly reduces the occurrence of non-predictable and non-controllable events. Nonetheless, some algorithms used in machine learning, as the genetic algorithm, are purposely based on the introduction of random events to better perform optimization tasks in extremely non-linear problems.

- **Language variety**. The use of natural language represents one of the crucial points of the adopted approach (cf. Table 18). As assumed in Fig. 28, the manipulation of the design concepts in form of language enables our brain to quickly generate different CVs by means of the KCAs (cf. Table 24). The more varied our lexicon (i.e. the more different words we know), the more different concepts we can express and combine.

  Brain. Estimating an average size of individual lexicon is particularly difficult. Based on [79–81], it mainly depends on age and education. As argued in [80], people learn and use thousands of inflected forms of words which however share the same core meaning. In the current analysis, we have thus adopted the concept of "word family" [7] (cf. Table 25) as measure of lexicon size, in accordance with [79]. Considering English, about 54,000 different word families exist [79]. On average, a 20-years-old native speaker knows about 11,100 word families (about 16,700 different words) [80,81]. A lexicon of 17,000 word families can be estimated for university graduates, while really cultured people meanly achieve 20,000 word families. It is however interesting to notice that we could comprehend the 95% of the oral English by knowing "only" 2,000-3,000 word families.

  Machine. The number of different words which can be stored in the knowledge base is potentially unlimited. Consequently, a machine can potentially manipulate and combine a larger variety of concepts than brain, thus increasing the possibility of generating breakthrough CVs.

- **Multidisciplinary abilities**. Based on Table 3, a broad multidisciplinary specialization increases the possibility of making analogies between different and apparently unrelated concepts, which favours the birth of breakthrough solutions.

  Brain. On average, we are specialized in one discipline, and we have superficial knowledge in the other ones. Designers are rarely specialized in two or more different disciplines. Due to this aspect, they often have difficulty in sharing their knowledge with others specialized in different disciplines. This is also linked to cognitive biases: we feel safe in our domain, and it is hard to admit different standpoints inherent to other disciplines (cf. Table 6).

  Machine. The number of disciplines a machine can specialize in is potentially unlimited: analogies between very different and apparently unrelated concepts are more likely. On the other hand, the number of disciplines depends on the number of programmers with different specializations who contribute to the knowledge base of the machine.

Regarding our purposes (cf. Table 9), almost all the aspects reported in Table 25 represent a limit for the brain. Notably, the first four ones are limits due to memory which cannot be consciously controlled and are almost impossible to eliminate, while those due to language variety and multidisciplinary abilities cannot be easily balanced. On the other hand, the emulation of the KCAs by means of the machine requires a non-negligible programming effort, and it thus involves a potential limit in the use

[7] A word family is a group of different words sharing the same core meaning (e.g. stimulate, stimulation, stimulated, stimulating, stimulates, stimulative, etc.). The first member represents the base form, while the following ones constitute the derived forms, which can be verbs, adjectives, nouns, etc. By knowing the meaning of either member of a family, one can easily infer the meaning of all the other members of the same family [79].

of the machine. In general, the comparison in Table 25 suggests that machine is potentially more suitable for conducting conceptual design according to our purposes.

2.1.2.3. Final considerations

Based on sections 2.1.2.1 and 2.1.2.2, two important considerations can be finally remarked:

- On one side, our brain is endowed with very powerful cognitive capabilities. As seen in Fig. 28, different design concepts can be quickly retrieved from our memory, and effectively manipulated by means of the KCAs (cf. Table 24) to develop novel CVs.
- On the other side, brain performances are limited by several aspects, which however do not affect the machine (cf. Table 25).

By means of machine learning, we want to exploit the power of the KCAs and all the advantages offered by machine during conceptual design, without being affected by brain limits. This is the idea on which the methodology of the developed demonstrator is based, and thus the motivation for using machine learning. As reported in Fig. 23, due to the difficulties encountered in realizing this ambitious idea, our efforts in this work will be mainly focused on the first step of the conceptual design (identification of the SPs, cf. Fig. 28). This means that the machine learning methods integrated in the developed demonstrator will mainly focus on emulating the KCAs of abstraction and generalization (step 1a, cf. Table 24). The second step of the conceptual design procedure (cf. Fig. 28) will be instead performed by the human user, and we are thus aware that the development of the CVs could be potentially influenced by memory biases and other brain limits (cf. Table 25). This important assumption will be resumed in chapter 3 before presenting the demonstrator.

2.2. Presentation of the implemented machine learning methods

In this section, we want to introduce and present the machine learning methods used in the current work. Machine learning, afterwards denoted as ML, generally involves the implementation of computational algorithms, able to emulate specific human cognitive abilities by learning from the surrounding environment. In the recent decades, many ML techniques have been developed for different purposes, as pattern recognition, data clustering and computer vision, and are more and more used in different domains as engineering, medicine, finance and entertainment [47]. Before presenting the organization of this section, let us introduce some important definitions regarding the used terminology.

- We refer to "ML model" (or "machine learning model") to indicate the type of human cognitive ability or a specific aspect of the human intelligence we want to emulate.
- We refer to "ML method" (or "machine learning method") to identify the computational procedures (i.e. the algorithms) which a given learning model is implemented with.

These definitions are summarised in Table 26.

Table 26: Definitions of machine learning model and machine learning method.

Terminology	Definition
ML model (or machine learning model)	It represents the type of human cognitive ability to be emulated
ML method (or machine learning method)	It represents the computational procedure (algorithm) by which a given learning model is implemented.

The current section is organized as follows. The three principal ML models, called supervised, unsupervised and reinforcement learning, are presented in section 2.2.1. The ML methods implemented in this work are based on the supervised and unsupervised learning models and are respectively presented in sections 2.2.2 and 2.2.3. Notably, our objective is to well outline their base ideas and to describe their functioning, highlighting the main parameters and their influence. This will enable non-expert readers to become more familiar with the principal technical aspects generally involved in ML algorithms. Some indications to lighten the reading of the following sections are rather provided to ML experts.

2.2.1. Learning models

Based on current established literature on machine learning and in particular on [47,50,82,83], three principal learning models exist: supervised learning, unsupervised learning and reinforcement learning, which are respectively presented in sections 2.2.1.1, 2.2.1.2 and 2.2.1.3. Their summary is then provided in section 2.2.1.4, together with some final considerations. The ML expert readers can directly skip to section 2.2.1.4.

2.2.1.1. Supervised learning

The base principle of supervised learning is to emulate the human generalization ability (cf. Table 23). As shown in Fig. 29a, the supervised learning model typically involves two distinct phases: training and generalization. Let us analyse these phases in detail according to [47,50,82,84].

Phase 1: training. This phase concerns the calibration of the learning algorithm on a suitable set (or array) of training samples (called training set), which represent the ensemble of knowledge that we want the machine to learn. According to Fig. 29a, each training sample consists of an input, called training input, and the corresponding output, called training output. Let us imagine, for instance, to train our learning algorithm for classifying dogs and cats into two different categories, labelled as “DOG” and “CAT”. The adopted training set is shown in Fig. 29b: each sample is characterised by an image of dog or cat (training input) and by the label of the category associated to the image (training output), i.e. what each image truly represents. With an iterative process, the training set is submitted to the learning algorithm, which is thus calibrated to associate the “DOG” label to dog images, and the “CAT” label to cat images (cf. Fig. 29a).

Phase 2: generalization. The trained algorithm is now able to recognize dogs and cats, by associating itself the label “DOG” or “CAT” to new images. The latter ones still depict dogs and cats, which are however different from the dogs and cats involved in the training set. As shown in Fig. 29a, the trained algorithm exploits the acquired knowledge to generalize (cf. Table 23), i.e. it can associate the same meaning (i.e. the same label) to new inputs which are similar to the training ones. The label associated to the new inputs is called generalization output (cf. Fig. 29a).

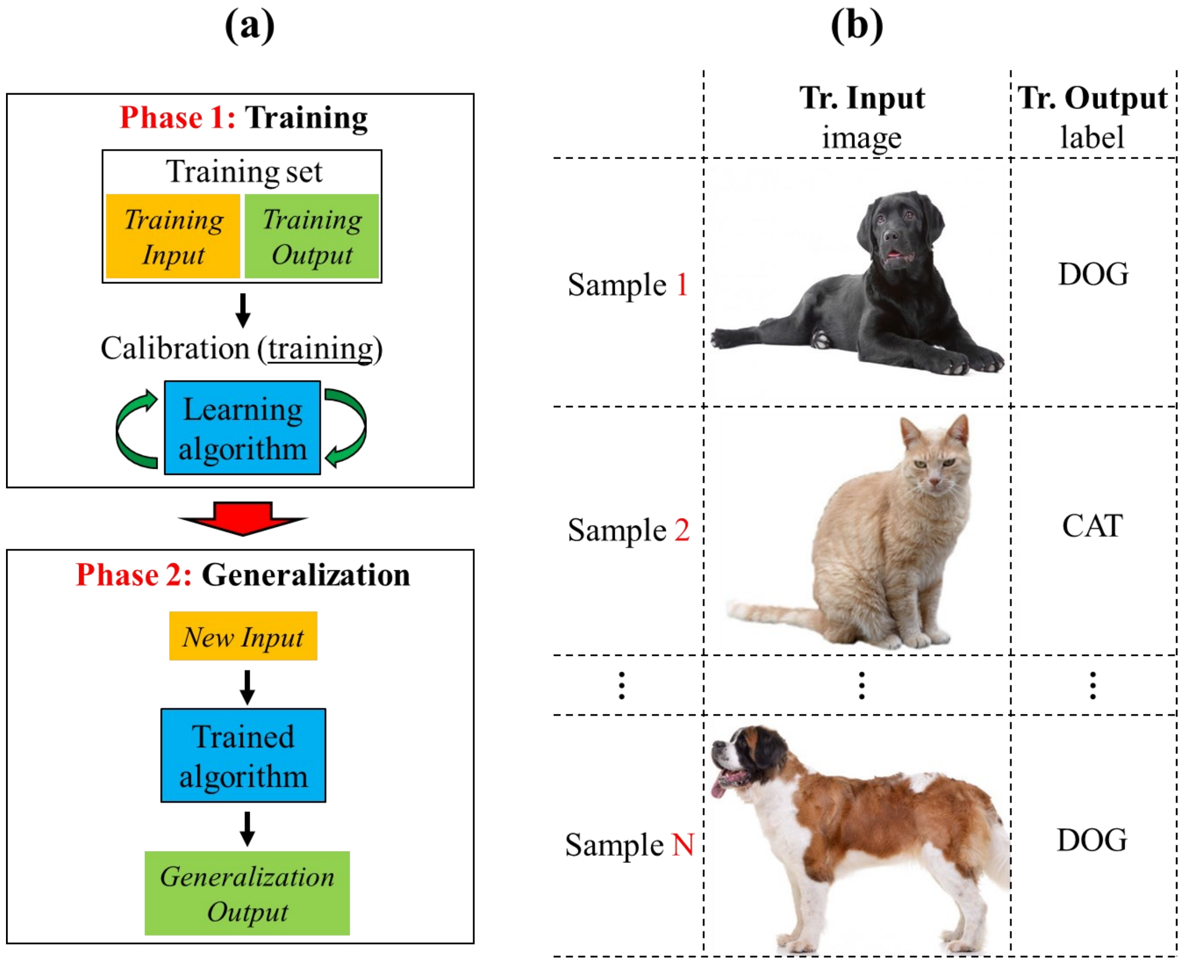


Fig. 29: (a) Scheme of the supervised learning model and (b) example of training set for the classification of dogs and cats.

If the submitted training set is not enough representative of the data to be classified, the trained algorithm will have poor generalization performances. Notably, the training outputs represent a sort of “absolute truth”, on which the algorithm is calibrated, and which enable to directly check if the resulting generalization outputs are pertinent or not. Due to this aspect, the supervised learning model is not strictly plausible from the biological standpoint (absolute truth does not exist in nature). However, supervised learning is very effective and very suitable for many practical tasks, mainly regarding:

- Classification, i.e. the object categorization based on the recognition of common systematic patterns or characteristics. Software of image recognition typically reposes on this kind of approach. In the example above (cf. Fig. 29b), dog and cat images can be classified according to different systematic characteristics, as size, ears’ shape, etc.

- Prediction, i.e. the approximation of a future event occurring in a known system. For instance, predictive maintenance software emploies this kind of approach: a given system (as an industrial machine) is continually monitored and the responses to future potential issues are elaborated in real time based on the collected data [50,84].

The supervised learning model is summarised in Table 28.

2.2.1.2. Unsupervised learning

The unsupervised learning model concerns the identification of hidden patterns or clusters (i.e. data groupings) in a set of samples, called training set. A cluster is defined as a group of training samples similar to one another. The samples contained in a cluster are thus dissimilar to the samples contained in other clusters. Contrary to supervised learning, the training set now consists of only unlabelled input data (without the corresponding output). As shown in Fig. 30a, the learning algorithm analyses the training inputs and groups them into different clusters which depend on the adopted clustering features, afterwards denoted as CFs [47,50,82,84].

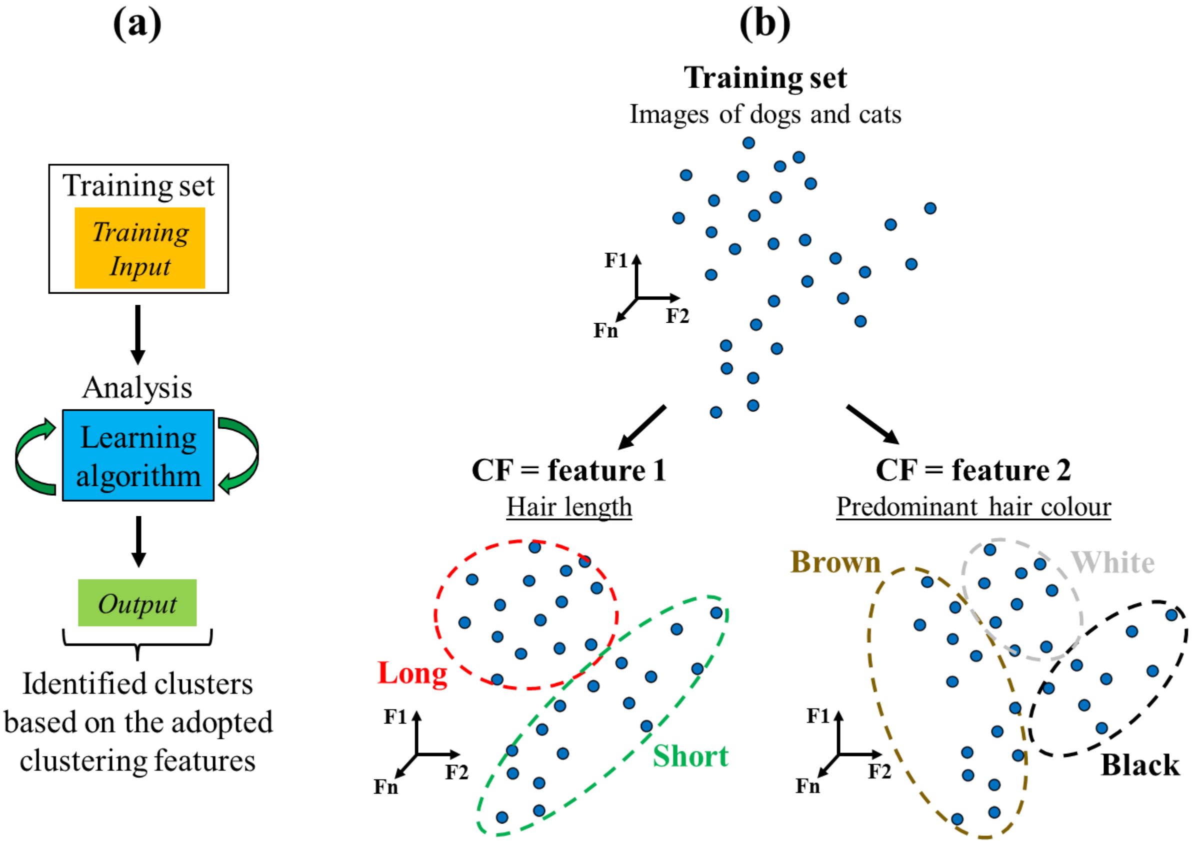


Fig. 30: (a) Scheme of the unsupervised learning model; (b) representation of the training set (dog and cat images, cf. Fig. 29b) in an n-dimensional space (upper part) and clustering of the training samples according to the adopted CFs (lower part).

Let us resume the example illustrated in section 2.2.1.1 to better illustrate the unsupervised learning model and the concept of clustering feature (CF). The considered training set corresponds to the unlabelled images of dogs and cats (only the training inputs, cf. Fig. 29b). As shown in Table 27, let us describe the training samples based on *n* different features, abstracted from the subjects of the studied images (cf. Fig. 29b). Each sample is thus expressed by means of different words, which represent the values of the abstracted features.

Table 27: Description of the training samples (cf. Fig. 29b) based on abstracted features.

	Feature 1	*Feature 2*	…	*Feature n*
Sample #	**Hair length**	**Predominant hair colour**	**…**	**…**
Sample 1	short	black	…	…
Sample 2	short	white	…	…
…	…	…	…	…
Sample N	long	brown	…	…

Table 27 provides an abstract representation of the training set: each feature can be isolated from the other ones and considered as a specific analysis object, emulating the abstraction ability of the brain (cf. Table 23). Moreover, as shown in Fig. 30b, the training set can be now represented as an ensemble of points in a n-dimensional space, where the dimensional axes (F1, F2, …, Fn) correspond to the abstracted features. The learning algorithm tries to group the training samples by "measuring" (or identifying) the similarities between their features. The two clustering examples proposed in Fig. 30b are characterised by two different CFs (feature 1 and feature 2, cf. Table 27), i.e. the features by which the samples are grouped. Notably, the CF corresponds to the "specific analysis object" in the definition of abstraction (cf. Table 23). In these cases, the similarity of the clustered samples is based on the words used in the description of the adopted CFs: for instance, the cluster "short" contains the samples where feature 1 is equal to "short", while the cluster "black" contains the samples where feature 2 is equal to "black". Depending on the adopted CFs, the same training samples can be clustered in different ways leading to different interpretations of the training set. Contrary to supervised learning, there is no "absolute truth" (training output) to directly check if results are pertinent or not. In general, it is thus preferred to verify a-posteriori if the generated clusters are representative of the chosen clustering features. Due to this aspect, the unsupervised learning is considered more biologically plausible than supervised learning. The practical uses of the unsupervised learning model mainly regard:

- Data organization and interpretation. Preliminary data clustering can be very effective to organize and better visualize data structure, before processing them by means of other methods.
- Anomaly detection, as in the fraud detection systems, for instance. An anomaly can be generally considered as a sample of the analysed training set which does not belong to anyone of the identified clusters. The concept of anomaly can be better understood with the help of Fig. 31: can you identify the "anomalous cat"? At first glance, we would probably answer that the black cat is the anomaly, because our attention is immediately captured by the colour difference. However, if we are interested in the direction of cats' look, the white cat in the centre will rather constitute the anomaly. In accordance with Fig. 30, anomaly identification thus strongly depends on the adopted CFs.

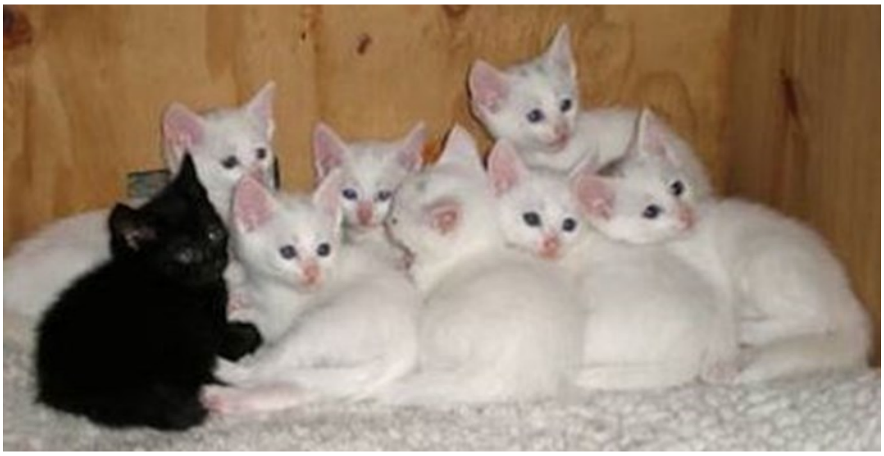


Fig. 31: Example of anomaly detection.

The unsupervised learning model is summarised in Table 28.

2.2.1.3. Reinforcement learning

The base idea of this model is to emulate a real time learning process conducted by trial and error, which is very similar to the way a baby learns [50]. The typical scheme of reinforcement learning is reported in Fig. 32, where the learning algorithm is "impersonated" by an agent exploring a given environment by attempts. At each attempt, the agent tries an action and immediately receives feedback from the environment. The feedback indicates whether the made action was right or wrong, and it consists of a "reward" or a "punishment", which reinforces the agent's knowledge of the environment (cf. Fig. 32). At the beginning, the agent has no knowledge of the environment, and he will thus receive many negative feedbacks for the first tried actions. However, after several attempts, the agent will gradually improve himself, by "calibrating" its behaviour to take only good decisions. Similarly to supervised learning (cf. Fig. 29a), actions and feedbacks can be respectively considered as training inputs and outputs (cf. Fig. 32). On the other hand, while in supervised learning the training set is entirely known from the beginning (Fig. 29a), in reinforcement learning the training inputs (actions) are generated in real time at each attempt, based on the training inputs (feedbacks) received in the previous attempts [47,50,83].

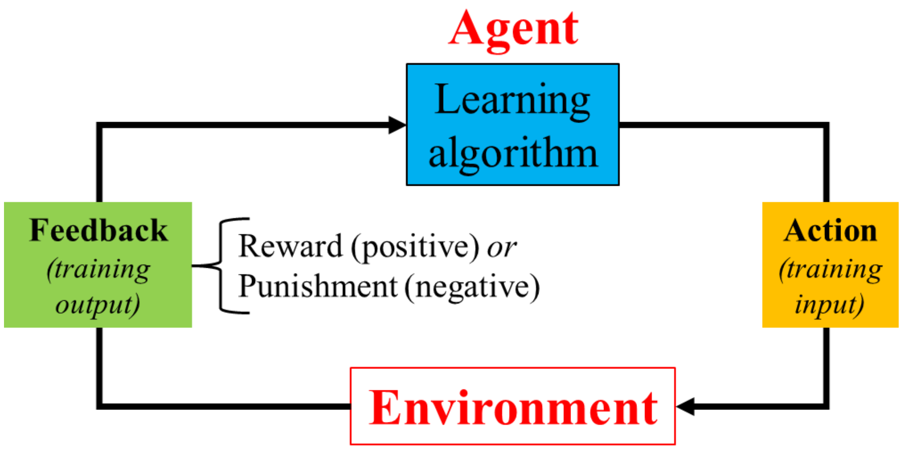


Fig. 32: Scheme of the reinforcement learning model.

Very common uses of reinforcement learning mainly regard:

- Robotics, where it is typically employed to teach machine different actions based on the surrounding environment. Many applications also involve automatic pilot devices for cars and drones [50].
- Adaptive online learning, where the knowledge of already trained systems continues to improve during their operation. This approach has been recently applied to videogame AIs, whose knowledge gradually adapts to the player's ability, in order to gradually increase game difficulty [83,85].

Let us see an example of reinforcement learning: let us imagine teaching a robot (agent) to walk, going as far as possible without damaging itself. The robot sees the environment through a camera. At each step (action), it receives images about its position and a feedback about its integrity, and it thus decides to make a new step (new action). The robot will encounter many obstacles du ring the walk. By crashing against obstacles, it will gradually learn to avoid them to continue advancing. At the beginning, the robot knows nothing, and has no danger instinct. If it walks on a cliff edge, it will need to fall off the cliff to learn it took a bad decision: too late, because it will be already destroyed. This can represent a big drawback regarding reinforcement learning. Indeed, it is preferred to preliminary train the agent on the basic environment laws by simulations, before passing to the real environment [47,50,83]. The reinforcement learning model is summarised in Table 28.

2.2.1.4. Summary and final considerations

The presented ML models are summarised in Table 28, reporting their base principles and main application tasks.

Table 28: Summary of the ML models: base principles and main application tasks.

ML model	Base principles	Main application tasks
Supervised learning	Emulation of the human generalization ability, involving two phases (cf. Fig. 29a). *Phase 1: training*. Calibration of the learning algorithm on the training set (i.e. the ensemble of knowledge to be learnt by the machine), to acquire the generalization ability. *Phase 2: generalization*. Exploiting the acquired knowledge to generalize (cf. Table 23). **Warning**: The training set must be sufficiently representative of the generalization database.	• Classification: categorization based on the recognition of common systematic patterns or characteristics (e.g. image recognition). • Prediction: approximation of future events occurring in a known system (e.g. predictive maintenance).
Unsupervised learning	Identification of clusters in a dataset (training set). *Definition of cluster*. Group of training samples similar to one another. The samples contained in a cluster are thus dissimilar to the samples contained in other clusters. The identified clusters depend on the adopted clustering features (CFs, i.e. the abstracted features by which the training samples are grouped, cf. Fig. 30b and Table 27). **Warning**: the pertinence of the identified clusters should be verified a posteriori.	• Data organization and interpretation: clustering to preliminary organize and better visualize data structure. • Anomaly detection: clustering data to identify anomalies according to different CFs (e.g. fraud detection systems).
Reinforcement learning	Emulation of a real time learning process conducted by trial and error. The agent (learning algorithm) explores the environment by attempts. Tried actions involve feedbacks from the environment (rewards or punishment, cf. Fig. 32), which enable agent to reinforce his knowledge in order to take only good decisions. **Warning**: preliminary training by simulation is suggested before passing to the real environment.	• Robotics: teaching machines different actions based on the surrounding environment (e.g. autopilot devices). • Adaptive online learning: Improving (adapting) the knowledge already trained systems (e.g. videogame AIs).

According to Table 28, these ML models represent the base for emulating many human cognitive abilities. As seen in section 2.2.1.1 and 2.2.1.2, the characteristics of the supervised and unsupervised learning models make them very suitable to emulate the KCAs of generalization and abstraction, i.e. those involved in the first step of the conceptual design procedure (cf. Fig. 28). According to section

2.1.2.3, these are the machine learning models on which the methodology of the current demonstrator is based.

2.2.2. The implemented supervised learning-based method: artificial neural networks (ANNs)

The two mainly used types of supervised learning algorithms are the artificial neural networks (ANNs) and the random forest (RF) [47,50,86–89]. The first type was chosen in this thesis. The implemented ANN will be employed for classification tasks, emulating the generalization ability of the human brain (cf. Table 28). In this section, we want to describe how data are processed in ANNs and how calibration (training, cf. Fig. 29a) is conducted, highlighting the influence of the main parameters. The current section is organized as follows. In section 2.2.2.1, ANNs and RF are introduced and compared, and the motivations for the choice of ANNs are provided. In section 2.2.2.2, the adopted type of ANN is presented and discussed in detail. The procedure to train ANN is introduced and presented in section 2.2.2.3. Some mathematical parameters regarding the training procedure are briefly discussed in section 2.2.2.4. The fundamental guide principles to improve training efficacy are provided in section 2.2.2.5. The implemented ANN is finally presented and tested in section 2.2.2.6. The experts of supervised learning and ANNs can directly skip to section 2.2.2.6.

2.2.2.1. Introduction and comparison of Artificial Neural Networks (ANNs) and Random Forest (RF)

As reported in [47,50,86–89], two different ML methods (or algorithms cf. Table 26) are mainly used to perform classification and prediction tasks (cf. Table 28): artificial neural network (ANN) and random forest (RF), respectively represented in Fig. 33a and Fig. 33b.

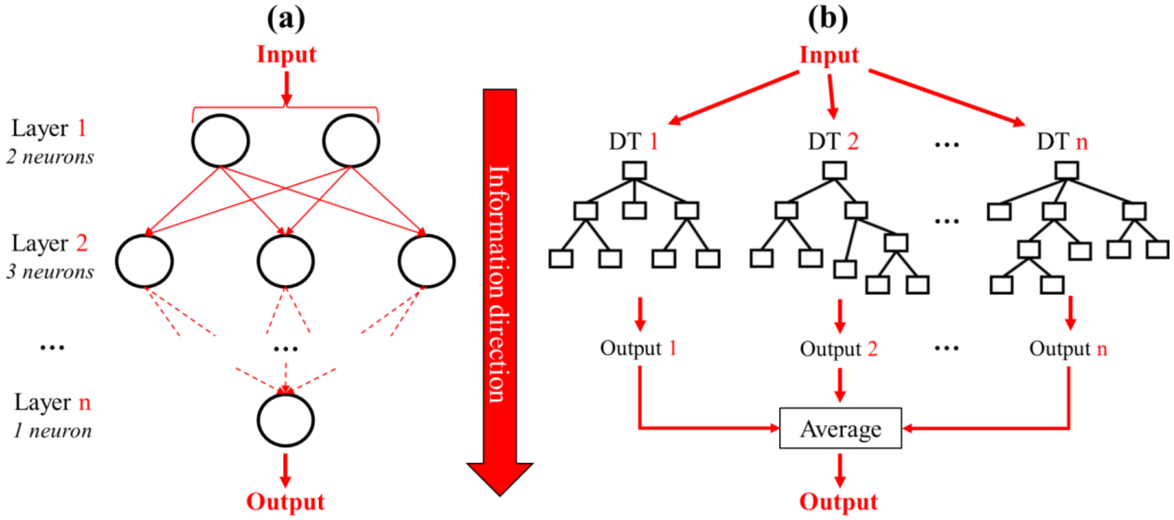


Fig. 33: (a) Representation of an artificial neural network (ANN) and (b) representation of a random forest (RF).

ANN and RF are based on different ideas. An ANN is inspired to biological neural systems and consists of multiple connected and interdependent artificial neurons, which are typically organized in different layers (cf. Fig. 33a). Input information is processed layer by layer by all neurons simultaneously, and the final classification (or prediction) output is provided by the neurons in the last layer. Instead, a RF is an ensemble of independent decision trees (DTs), each one with a different structure [90]. As shown in Fig. 33b, each DT in the RF elaborates the input information and provides its own classification (or prediction) output. Therefore, the RF final output is the average of all the outputs given by the single DTs. The ANN-RF comparison proposed in Table 29 helps to understand which method is more suitable to our purposes. According to [86–89], the comparison is based on three main criteria, and the characteristics highlighted in green indicate which method is better regarding a given criterion.

Table 29: Comparison ANN-RF (for green colour see text description) [50,86–89,91–93].

Comparison criterion	ANN	RF
Type of processable data	Numerical, text (natural language), audio and image	Numerical
Robustness of the method to dispersed training data (influence on the accuracy of the trained algorithm)	High	Low
"Tuning parameters" (or hyperparameters, influence on the accuracy of the trained algorithm)	Many and less user-friendly	Few and more user-friendly

The first criterion suggests that ANNs can be used to process any type of data, including natural language, whose processing is one of the main PAAs (cf. Table 18), while RF based methods are typically limited to process numerical data. The robustness (second criterion, cf. Table 29) measures the attitude of ML methods to not be influenced by dispersed (anomalously distributed) training data sets, which is often the case of natural language processing. Based on [86–89], this characteristic influences the accuracy of the trained algorithm. As indicated in Table 29, ANNs are less influenced by dispersed data than RFs. The third comparison criterion regards the number and the entity of the "tuning parameters" (often referred to as hyperparameters) of the learning algorithm. These parameters are used to regulate the training phase, and they should be set-up and tuned during the training phase (cf. Fig. 29a) to ensure results relevancy and improve the accuracy of the used ML method [91]. In general, the hyperparameters involve the algorithm architecture, as the number of neurons and layers for ANNs and the number of DTs for RFs, and other different mathematical parameters. As indicated in Table 29, RF has fewer hyperparameters, which are also more user-friendly than ANN [86–89]. In summary, ANNs can be employed in more cases, including language processing (PAAs, cf. Table 18), and are more robust to dispersed training data, even if their use generally requires a deeper study than RFs. Otherwise, RFs are more interesting for fast numerical applications where a deep knowledge of machine learning is not required. Regarding ANNs, many indications to help users setting-up hyperparameters are however provided in literature [50,91–93]. For these reasons, ANNs represent the supervised learning method implemented in this work.

2.2.2.2. Adopted type of ANN and information processing in neurons

Based on Fig. 33a, the type of ANN adopted in this work is shown in Fig. 34, and it is characterised by three layers: input layer, hidden layer and output layer. Neurons are denoted with a letter, which indicates the respective layer, and a number. For example, "i2" is the second neuron of the input layer, "h3" is the third neuron of the hidden layer, and "o1" is the first neuron of the output layer (cf. Fig. 34). The numbers of input and output neurons depend on the dataset to be processed (i.e. on the studied problem), while the number of hidden neurons is an hyperparameter (cf. Table 29) and it should be selected by attempts during the training phase [50,91–93]. The chosen ANN model also involves two additional “special neurons”, called *bias*[8] neurons and denoted with “b” (cf. Fig. 34). In each layer, each neuron is connected to all the neurons of the following layer by means of artificial synapses. Each artificial synapse is identified by a weight, denoted with “w”, i.e. a real number (R) whose function is inspired to the adjustability of the biological chemical synapses. As reported in Table 19, this property is the base of the learning process in biological neural systems, and it consists of varying the quantities and the nature of the neurotransmitters released across the synaptic clefts (cf. Fig. 24b). In a similar way, the ANN learning process is based on the calibration (adjustment) of the synaptic weights during the training phase (cf. Fig. 29a). In mathematical terms, the synaptic weights thus represent the system unknowns. Regarding the annotations used in Fig. 34, the superscript of the synaptic weights indicates the connected layers, while the subscript indicates the numbers of the connected neurons. For example, the synapse w_{13}^{ih} connects the neurons “i1” and “h3”, the synapse w_{22}^{ho} connects the neurons “h2” and “o2”, and the synapse w_{3}^{bh} connects “h3” to the respective *bias* neuron (cf. Fig. 34).

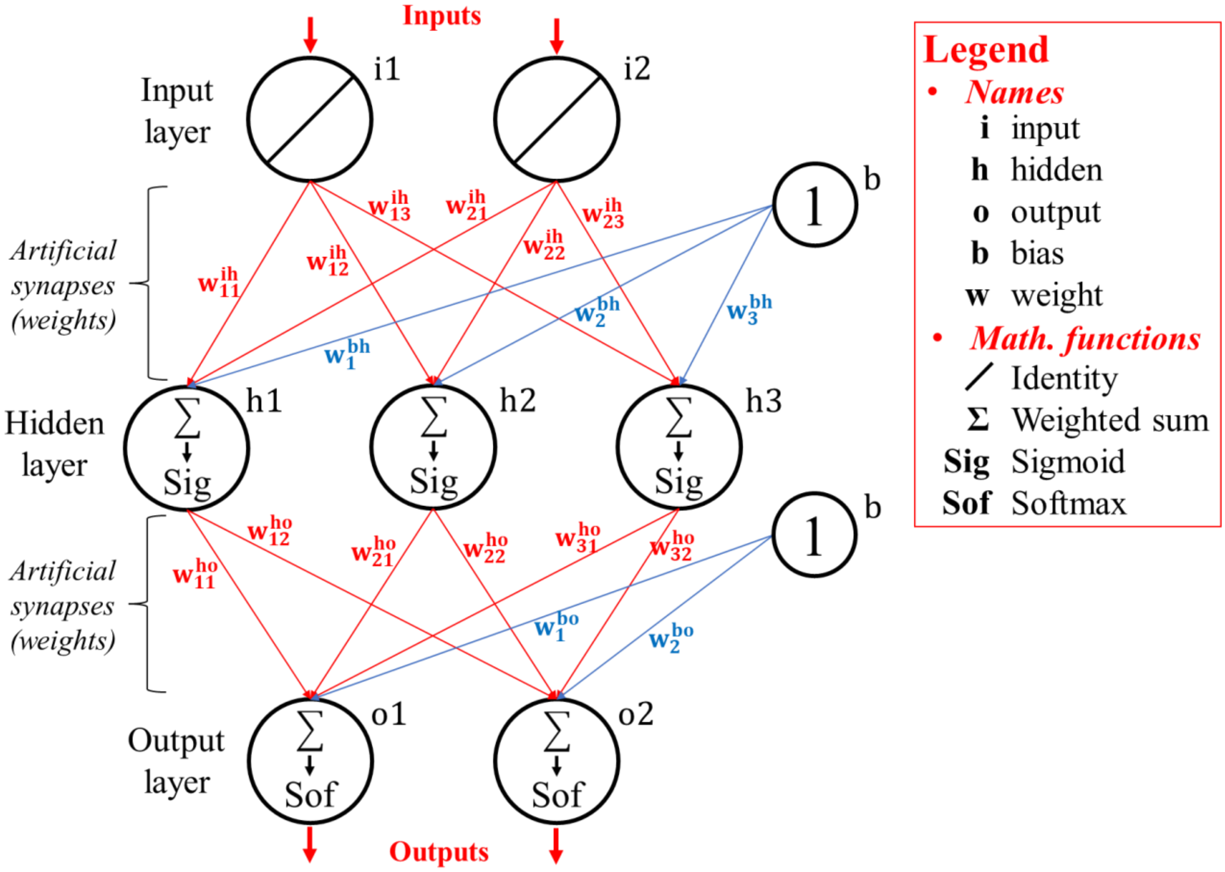


Fig. 34: Type of ANN adopted in the current work.

Each artificial neuron represents an elementary processing unit, where received information is elaborated by means of different mathematical functions and then provided to the neurons of the following layer. The mathematical functions contained in the neurons are reported in Fig. 34 according to [50]. Their role can be better understood by illustrating in detail how information is processed layer by layer, assuming the ANN as already trained (the synaptic weights are calibrated). For each layer, let

[8] In this context, the term “bias” is used for mathematical purposes [50]. There is no connection between bias neurons and memory biases (cf. Table 22 and Table 25).

us adopt "x" to indicate the input of the generic neuron, and "y" to indicate the output of the generic neuron.

Input layer. The role of input neurons is to introduce the input data provided by user into the ANN. As shown in Fig. 34, each input neuron is characterised by the identity function (y = x), which means the neuron output y, i.e. the information passed forward to the hidden layer, is equal to the neuron input x, i.e. the information provided by user (x).

Hidden layer. Let us see how information is processed by hidden neurons. In Fig. 35a, the neuron h1 is considered. Information processing involves two steps, characterised by two different functions: sum and sigmoid (cf. Fig. 34). As shown in Fig. 35a, the first step concerns the accumulation of the information incoming from the input neurons by means of a weighted sum, generating the accumulated signal x'. This operation is inspired to the accumulation of input signals in a unique nucleus potential which occurs in a biological neuron (cf. section 2.1.1.1), as indicated in Fig. 35b.

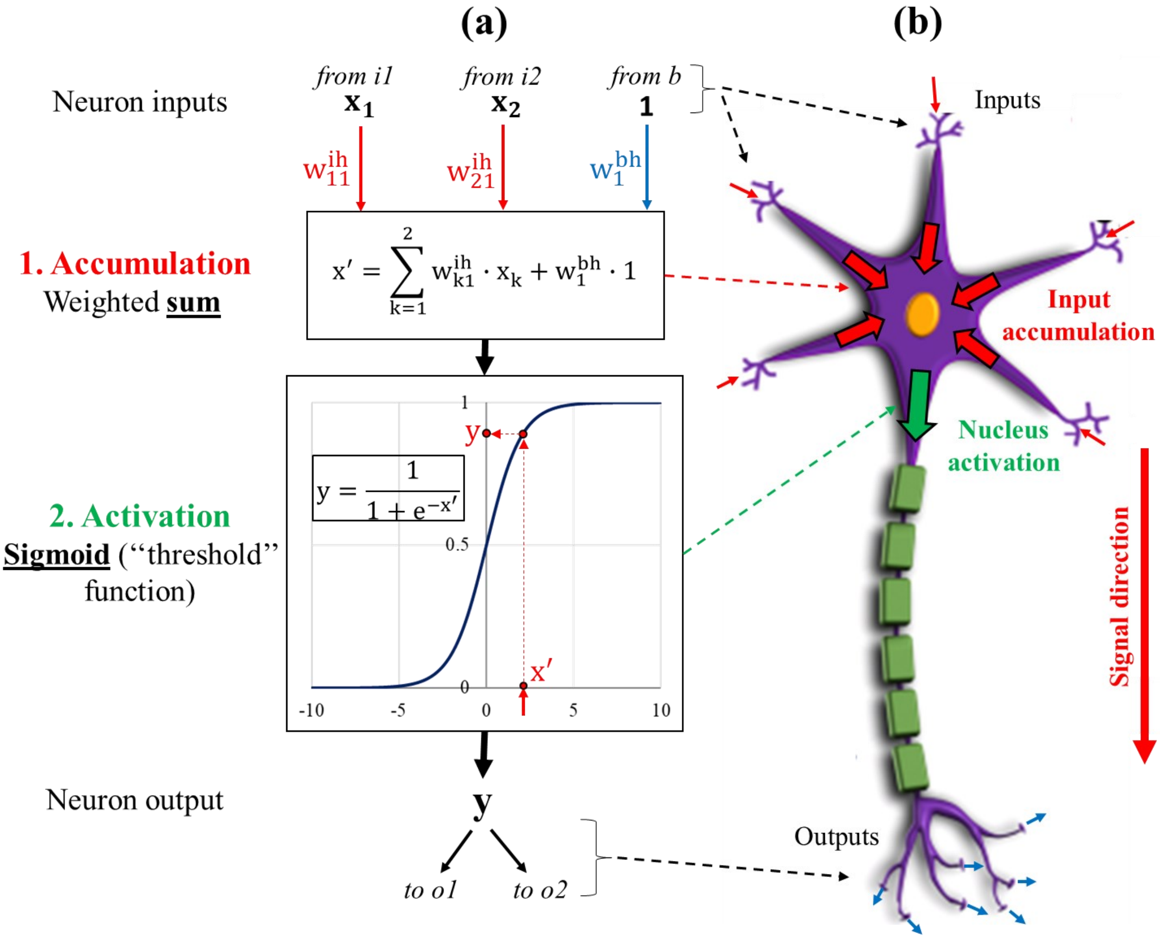


Fig. 35: (a) Information processing in the artificial neuron h1 (cf. Fig. 34); (b) analogy with biological neuron (cf. Fig. 24a).

As shown in Fig. 35a, the second step involves the neuron activation and consists of passing the accumulated signal x' into the sigmoid function to generate the neuron output y, which is limited in the range between 0 (inhibiting) and 1 (stimulating). According to Fig. 35b, the base idea of using sigmoid function is to emulate the nucleus of a biological neuron, which activates if the accumulated signal exceeds a specific threshold potential (cf. section 2.1.1.1). The same operations of accumulation and activation are performed in the neurons h2 and h3. The outputs generated by the hidden neurons are then passed forward to the output layer (cf. Fig. 35a).

Output layer. As the hidden neurons, also output neurons involve the two sequential steps of accumulation and activation (cf. Fig. 35a). The accumulated signals x'_1 and x'_2 are first generated in the output neurons o1 and o2 by means of the weighted sum, in the same way as Fig. 35a. The activation of the output neurons is thus represented in Fig. 36, where the softmax function is used to generate the ANN outputs y_1 and y_2. As shown in Fig. 36, the softmax is a normalized exponential function and it represents a generalization of the Sigmoid function for multiple dimensions. The reason for using the sigmoid function in the output layer is explained as follows.

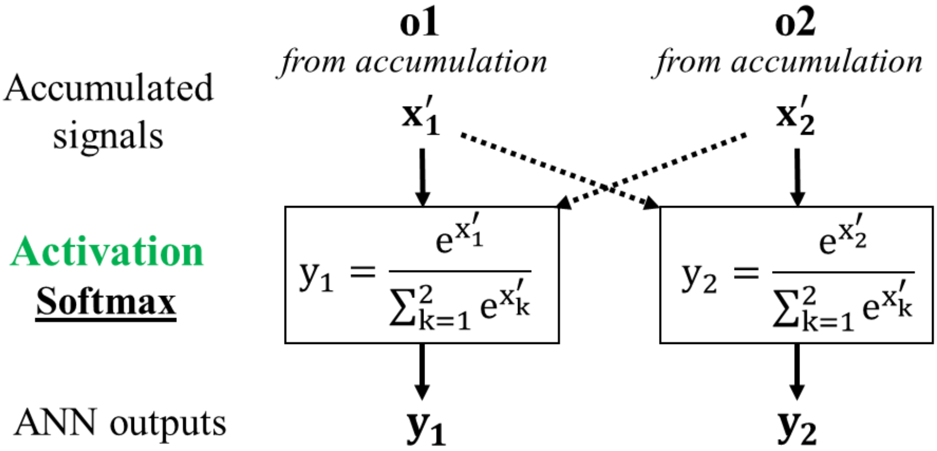


Fig. 36: Activation of the output neurons o1 and o2 (cf. Fig. 34).

Three different examples of softmax calculation are proposed in Table 30 for neurons o1 and o2, to better understand what kind of result is provided by this function. For the rows o1 and o2 (cf. Table 30), the values of x′ are chosen arbitrarily and the values of y (neuron outputs) are calculated with the equations reported in Fig. 36, while the row o1+o2 (cf. Table 30) provides the sum of x′ and y values.

Table 30: Examples of softmax calculation.

	Example 1		*Example 2*		*Example 3*	
Neuron	**x′**	**y = sof(x′)**	**x′**	**y = sof(x′)**	**x′**	**y = sof(x′)**
o1	3	0.73	-1	≈ 0	-4	0.95
o2	2	0.26	5	≈ 1	-7	0.05
o1+o2	5	1	4	1	-13	1

As shown in Table 30, the single neuron outputs (y) are limited in the range [0;1], and their sum is always equal to 1 (this is true even with more than two output neurons). For these reasons, the output of softmax function can be seen as a probability distribution in percentage, and it is thus very suitable for multiclass classification problems [94], as in our case.

The two similitude aspects identified in this section between ANN and biological neural systems are reported in Table 31, while the role of the mathematical functions used in the adopted type of ANN (cf. Table 30) is summarised in Table 32.

Table 31: Similitude aspects between ANN and biological neural systems (cf. Fig. 35 and Fig. 36) [47,50].

Aspect	**Biological neural system**	**ANN**
Learning process	Adjustment of the chemical synapses (cf. Table 19)	Calibration (adjustment) of the synaptic weights (w, cf. Fig. 34) during the training phase
Two-step information processing in neurons	1. Accumulation of input signals in a unique nucleus potential 2. Activation of the neuron generating an electrical output impulse (cf. section 2.1.1.1)	1. Accumulation of the neuron inputs into the signal x′ (cf. Fig. 35a) 2. Activation of the neuron generating the output y (hidden and output neurons, cf. Fig. 35a and Fig. 36)

Table 32: Role of the mathematical functions used in the ANN (cf. Fig. 34).

Layer	Step	Used function	Role
Input		Identity	Introducing the input data provided by user into the ANN
Hidden	1. Accumulation	Weighted sum	Accumulating (collecting) the information incoming from the input neurons
	2. Activation	Sigmoid	Generating a neuron output limited between 0 (inhibiting) and 1 (stimulating), similarly to the threshold potential of biological neurons.
Output	1. Accumulation	Weighted sum	Accumulating (collecting) the information incoming from the hidden neurons
	2. Activation	Softmax	Providing a multiclass probability distribution in percentage [%]

2.2.2.3. ANN training: introduction and procedure flowchart

The overall knowledge stored in our memory is encoded in the synaptic traces formed and adjusted during the processes of rehearsal (cf. Table 21). Similarly, the knowledge of an ANN is encoded in the synaptic weights calibrated during the training phase (cf. Table 31). Based on Fig. 29a, this phase notably consists of minimizing the error between the ANN outputs and the desired (target) outputs, i.e. the outputs of the samples contained in the training set (cf. Fig. 29b), by adjusting the synaptic weights [50]. The following example will help to better illustrate the training phase. The training set reported in Table 33 is assumed as example to illustrate the training phase, and it is composed by 100 training input and the 100 corresponding training outputs (cf. Fig. 29b). The idea is to teach the ANN in Fig. 34 to classify XY coordinates according to the four quadrants of the Cartesian plane. It is important to remark that the number of inputs for each training samples corresponds to the number of input neurons (i.e. two, cf. Fig. 34), and that the number of outputs for each training sample is equal to the number of output neurons (i.e. two, cf. Fig. 34). X and Y values are assumed limited in the range [-100;100]. Moreover, based on Table 32, training outputs are in form of probability arrays: for instance, sample 1 has 0% probability (t_1 = 0) to belong to the first or to the third quadrant (I | III, cf. Table 33), and 100 % probability (t_2 = 1) to belong to the second or to the fourth quadrant (II | IV, cf. Table 33).

Table 33: Training set assumed as example to illustrate the training phase: classification of the XY coordinates according to the four quadrants of the Cartesian plan.

	Training input **Coordinates**		*Training output* **Probability**	
# Sample	**X**	**Y**	**I | III (t_1)**	**II | IV (t_2)**
1	+30	-67	0	1
2	+58	+39	1	0
…	…		…	…
100	-42	-84	1	0

The flowchart of the iterative training procedure is schematized in Fig. 37a. Before starting the procedure, all synaptic weights ($w^{ih}, w^{ho}, w^{bh}, w^{bo}$, cf. Fig. 34) are initialized with random real numbers, as suggested in [50]. After starting the procedure, the training samples are presented to the ANN one by one. For each sample, two main steps can be distinguished: a forward step and a backward step.

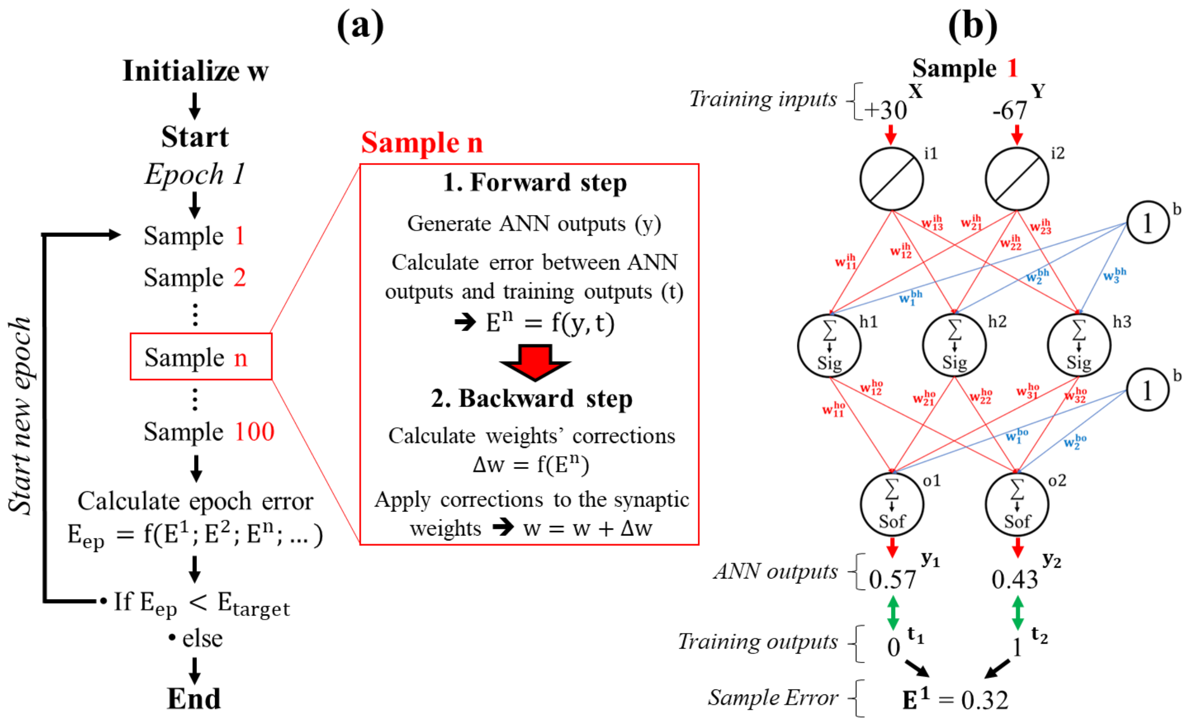

(a)
Initialize w
Start
Epoch 1
Sample 1
Sample 2
Sample n
Sample 100
Calculate epoch error
$E_{ep} = f(E^1; E^2; E^n; \dots)$
• If $E_{ep} < E_{target}$
• else
End
Start new epoch
Sample n
1. Forward step
Generate ANN outputs (y)
Calculate error between ANN outputs and training outputs (t)
➔ $E^n = f(y, t)$
2. Backward step
Calculate weights' corrections
$\Delta w = f(E^n)$
Apply corrections to the synaptic weights ➔ $w = w + \Delta w$
(b)
Sample 1
Training inputs
+30 X
-67 Y
i1
i2
b
1
h1
h2
h3
Sig
Sof
o1
o2
w^{ih}_{11} w^{ih}_{12} w^{ih}_{13} w^{ih}_{21} w^{ih}_{22} w^{ih}_{23}
w^{bh}_{1} w^{bh}_{2} w^{bh}_{3}
w^{ho}_{11} w^{ho}_{12} w^{ho}_{21} w^{ho}_{22} w^{ho}_{31} w^{ho}_{32}
w^{bo}_{1} w^{bo}_{2}
ANN outputs
0.57 y_1
0.43 y_2
Training outputs
0 t_1
1 t_2
Sample Error
$E^1 = 0.32$

Fig. 37: (a) Flowchart of the iterative training procedure; (b) forward step of the sample 1 (cf. Table 33).

Let us describe these steps.

Forward step. As reported in Fig. 37a, the nth training sample is first passed into the ANN. The sample error E^n is thus calculated with Eq. (7) (cf. [50]), as function of the generated ANN outputs (y) and of the training outputs (t_1 and t_2, cf. Table 33), i.e. the target outputs.

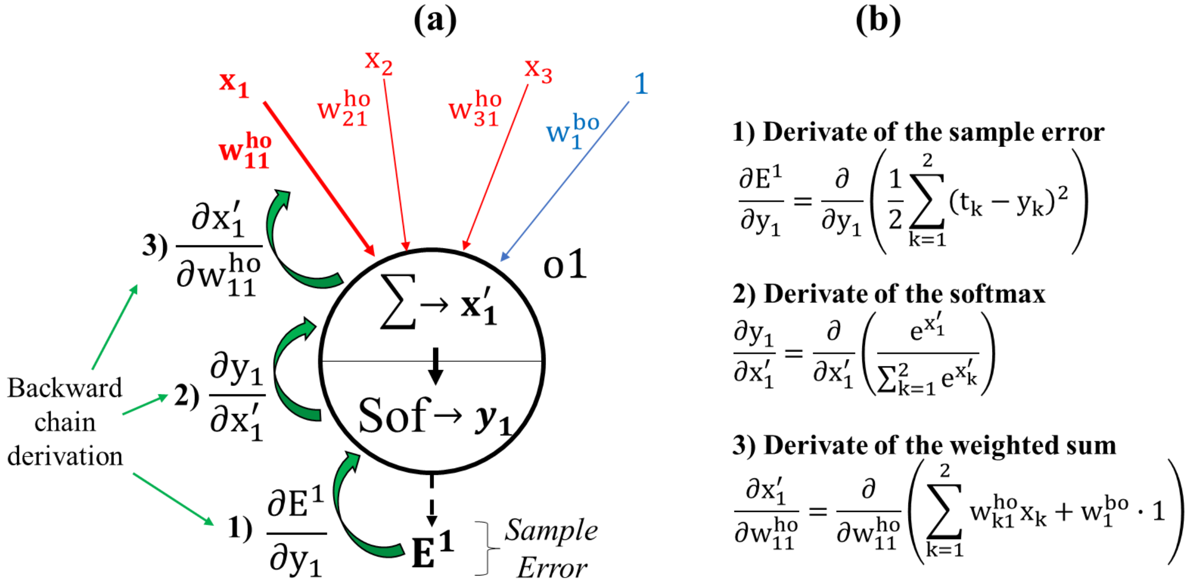
$$E^n = \frac{1}{2}\sum_{k=1}^{2}(t_k - y_k)^2. \qquad (7)$$

The forward step of the sample 1 (cf. Table 33) is represented in Fig. 37b. According to the response provided by the ANN, sample 1 has a 57% probability ($y_1 = 0.57$) to belong to the first or to the third quadrant (I | III, cf. Table 33), and a 43% probability ($y_2 = 0.43$) to belong to the second or to the fourth quadrant (II | IV, cf. Table 33). This leads to the sample error $E^1 = 0.32$, calculated with Eq. (7). It can be moreover remarked that the sum of y_1 and y_2 is equal to 1, in accordance with Table 30.

Backward step. Synaptic weights' corrections (Δw) are calculated as function of E^n (cf. Fig. 37a). The popular computational procedure used to perform this step is called back-propagation algorithm (BPA) [50,95,96]. Since the BPA is a well consolidated and accessible method, we will only provide some base passages to help understanding its functioning, without presenting the complete mathematical procedure implemented, which is however well documented in [50]. For instance, let us focus on the synaptic weight w^{ho}_{11}, connecting the neurons h1 and o1 (cf. Fig. 34). Let us calculate the correction associated to w^{ho}_{11}, i.e. Δw^{ho}_{11}, by resuming the training procedure from the calculation of the sample error E^1 in Fig. 37b. According to the BPA method, Δw^{ho}_{11} is expressed by the gradient of the sample error E^1 with respect to w^{ho}_{11}, as reported in Eq. (8):

$$\Delta w^{ho}_{11} = \frac{\partial E^1}{\partial w^{ho}_{11}} = \frac{\partial E^1}{\partial y_1}\cdot\frac{\partial y_1}{\partial x'_1}\cdot\frac{\partial x'_1}{\partial w^{ho}_{11}}, \qquad (8)$$

where the operator "∂" indicates the partial derivate. Based on [50], the gradient in the second term of Eq. (8) can be decomposed into three components by means of the derivative chain rule[9]. By focusing on the neuron o1, Fig. 38a shows the process of backward chain derivation, while Fig. 38b reports the expressions of the three gradient components, which correspond to the partial derivates of the sample error (cf. Eq. (7)), of the softmax function (cf. Fig. 36), and of the weighted sum (cf. Fig. 35a).


(a)
(b)
x_1
x_2
x_3
1
w_{11}^{ho}
w_{21}^{ho}
w_{31}^{ho}
w_1^{bo}
o1
$\sum \rightarrow x_1'$
$\text{Sof} \rightarrow y_1$
E^1
Sample Error
Backward chain derivation
3) $\frac{\partial x_1'}{\partial w_{11}^{ho}}$
2) $\frac{\partial y_1}{\partial x_1'}$
1) $\frac{\partial E^1}{\partial y_1}$

1) Derivate of the sample error

$$\frac{\partial E^1}{\partial y_1} = \frac{\partial}{\partial y_1}\left(\frac{1}{2}\sum_{k=1}^{2}(t_k - y_k)^2\right)$$

2) Derivate of the softmax

$$\frac{\partial y_1}{\partial x_1'} = \frac{\partial}{\partial x_1'}\left(\frac{e^{x_1'}}{\sum_{k=1}^{2} e^{x_k'}}\right)$$

3) Derivate of the weighted sum

$$\frac{\partial x_1'}{\partial w_{11}^{ho}} = \frac{\partial}{\partial w_{11}^{ho}}\left(\sum_{k=1}^{2} w_{k1}^{ho} x_k + w_1^{bo} \cdot 1\right)$$

Fig. 38: (a) Backward chain derivation in the neuron o1 (cf. Fig. 34) according to the BPA; (b) expressions of the gradient components (cf. Eq. (7) , Fig. 36 and Fig. 35a).

All the variables contained in the equations of Fig. 38b are known from the forward step (cf. Fig. 37b). Consequently, all the three gradient components can be easily calculated, and thus replaced in Eq. (8) to obtain Δw_{11}^{ho}. The corrections associated to the synaptic weights w^{ih} (cf. Fig. 34) are calculated in a similar way. In this case, the gradient components of Δw^{ih} corrections will be expressed as function of the accumulated signals (x') and outputs (y) generated in the hidden neurons[10]. As reported in Fig. 37a, the calculated corrections are finally applied to the synaptic weights, and the training procedure continues with the sample 2.

Forward and backward steps are performed for each training sample. Once all the 100 samples have been treated, the first iteration (epoch 1, cf. Fig. 37a) is completed. As shown in Fig. 37a, an epoch error E_{ep} is thus calculated to measure the efficiency of the training process, i.e. the classification accuracy achieved by the trained ANN. Typically, E_{ep} is calculated as the average error, the root mean square error or the maximum of the sample errors calculated during the just past epoch. However, in our case, the ANN outputs can be more easily interpreted as "right" or "wrong", depending on the resulting probability distribution. According to [50], the output of the ANN is considered right if the maximum y (between y_1 and y_2, cf. Fig. 37b) corresponds to the training output (t_1 or t_2, cf. Fig. 37b) which is equal

[9] Mathematical demonstrations and examples are reported at:

- https://tutorial.math.lamar.edu/classes/calci/chainrule.aspx
- https://en.wikipedia.org/wiki/Chain_rule

[10] A step-by-step back-propagation example which illustrates in detail the calculation of Δw^{ih} is reported at https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

to 1[11]. Otherwise, the output of the ANN is considered wrong. For example, the response provided by the ANN in Fig. 37b is wrong, since the maximum y ($y_1 = 0.57$) does not correspond to $t_2 = 1$. The epoch error E_{ep} is thus calculated with Eq. (9):

$$E_{ep} = 1 - \frac{N^R}{N^S}, \tag{9}$$

where N^R is the number of right responses provided by the ANN during the just past epoch, and N^S is the number of training samples in the training set. The lower E_{ep}, the more accurate the trained ANN. After calculating E_{ep}, the latter is finally compared to a target error E_{target} set by the user (cf. Fig. 37a). If $E_{ep} < E_{target}$ a new epoch is started. Otherwise, the training process ends. In general, the decreasing of E_{ep} epoch after epoch indicates that the training process is well performing (ANN classification accuracy is increasing).

2.2.2.4. ANN training: brief discussion on mathematical hyperparameters

In section 2.2.2.3, the training process of an ANN has been presented with the help of Fig. 37a. For brevity, some aspects regarding the calculation of weights' corrections (Δw, cf. Fig. 37a) with the BPA (cf. Eq. (8)) have been voluntarily omitted. Notably, this operation involves two "mathematical" hyperparameters (cf. Table 29), called learning rate (η) and momentum (μ), which are both defined in the range [0;1]. According to [50,95,96], Eq. (8) can be rewritten into Eq. (10) for the nth training sample:

$$\Delta w(\tau) = -\eta \frac{\partial E^n}{\partial w} + \mu \Delta w(\tau - 1), \tag{10}$$

where τ indicates the actual epoch, and the sample error gradient is calculated with Eq. (8), by means of the derivative chain rule (cf. Fig. 38). Based on Eq. (10), the role of learning rate (η) is to reduce the entity of the sample error gradient (i.e. $\frac{\partial E^n}{\partial w}$), while the role of momentum (μ) is to weight the correction of the actual epoch (i.e. $\Delta w(\tau)$) on the correction calculated in the previous epoch (i.e. $\Delta w(\tau - 1)$). In both cases, the main purpose is to slow down the learning process, to avoid the ANN from forgetting (losing) the knowledge acquired in the previous epochs and to prevent the learning algorithm from being trapped in local minima. The roles and the purposes of these hyperparameters are summarised in Table 34 according to [50,97]. Typically, the values of η and μ should be tuned with different training tests. In this work, these hyperparameters will be set-up at the beginning of the training phase, based on some indications reported in literature (cf. [50,91–93,97]) and on priorly conducted training tests (which are not presented here for brevity).

[11] This assures that all the other neuron outputs are lower, since the sum of all y outputs is always equal to 1 (cf. Table 30 and Fig. 37b).

Table 34: Roles and purposes of learning rate and momentum [50,97].

ANN math. hyperparameter	Role	Purposes
Learning rate (η)	Reduce the entity of the sample error gradient ($\frac{\partial E^n}{\partial w}$, cf. Eq. (10)) calculated with the derivative chain rule (cf. Eq. (8))	Slow down the learning process for: • Avoiding the knowledge acquired in the previous epochs to be forgotten (lost) by the ANN. • Preventing the learning algorithm (cf. Fig. 37a) from being trapped in local minima during the training phase.
Momentum (μ)	Weight the correction of the actual epoch ($\Delta w(\tau)$, cf. Eq. (10)) on the correction calculated in the previous epoch ($\Delta w(\tau - 1)$, cf. Eq. (10))	

We also believe that a more detailed mathematical explanation of these hyperparameters is not mandatory for the comprehension of ANN functioning in the context of this work. We thus address the reader to some literature contributions for their exhaustive discussion[12] [97–100].

2.2.2.5. ANN training: fundamental guide principles

The fundamental guide principles to improve the efficiency of the training process are provided in Table 35 according to [47,50,96]. The potential risks in not following the guide principles are also reported.

Table 35: Fundamental guide principles to train an ANN [47,50,96].

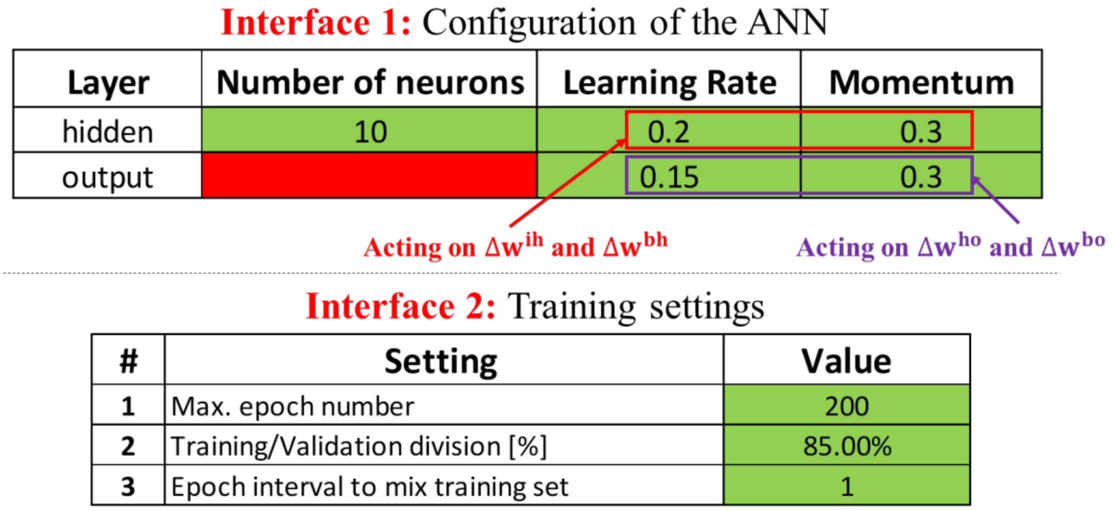
Guide principle	Potential risk if not followed
1. Use a high number of training samples (a thousand of sample at least is typically suggested).	The number of samples is not sufficient to allow the ANN acquires the generalization ability. The ANN could merely memorize the training samples (overfitting), obtaining a low classification accuracy.
2. All the used training samples must be different from each other.	The ANN knowledge is biased towards the repeated samples.
3. Mix the order of the training samples at each epoch.	The ANN merely memorizes the order of the training samples (overfitting), without acquiring the generalization ability (low classification accuracy).
4. At the end of each epoch, test the ANN with a validation set different from the training set: validation samples are not directly used to calibrate the synaptic weights with the BPA, but they are used to test the generalization ability acquired by the ANN during the training process. Epoch errors to be calculated: E_{ep}^{t} ➔ epoch error due to training samples E_{ep}^{v} ➔ epoch error due to validation samples	During the training phase, it is more difficult to check whether the ANN is effectively acquiring generalization ability (classification accuracy is increasing) or it is merely memorizing the training samples (overfitting).

[12] Some useful graphic explanations for setting-up learning rate and momentum are reported at:
- https://cnl.salk.edu/~schraudo/teach/NNcourse/momrate.html.
- https://www.jeremyjordan.me/nn-learning-rate/.
- https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d.

Based on Table 35, the mere memorization of the training samples (implying a poor classification accuracy of the ANN) is one of the most probable negative effect if the given guide principles are not observed. As argued in [47,50], the phenomenon of sample memorization is called overfitting. The first guide principle represents one of the main weaknesses of ANNs. Indeed, training data are not always readily available and their number is often limited, mostly for engineering applications where training samples derive from experimental data. In the following chapter, we will use a clever method to solve this problem and generate thousands of different training samples. Based on the fourth guide principles, validation samples are often used to test the generalization ability of the implemented ANN during the training phase. Contrary to training samples, validation samples are not involved in the weights' calibration process with the BPA (cf. Eq. (8) and Fig. 38). According to Table 35, the ANN classification accuracy is thus measured by calculating two different epoch errors at the end of each epoch: $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$, due to training samples, and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$, due to validation samples. This aspect will be further clarified in section 2.2.2.6.

2.2.2.6. Presentation and testing of the implemented ANN

Based on the theory presented in sections 2.2.2.2, 2.2.2.3 and 2.2.2.4, a versatile model of ANN integrating the classical BPA (cf. [95,96]) has been implemented in MATLAB [48]. The set-up interface of our ANN is entirely handled by means of the two Excel tables shown in Fig. 39, which make the parameter initialization very intuitive and original. The data contained in the green cells can be modified by the user.



Interface 1: Configuration of the ANN

Layer	Number of neurons	Learning Rate	Momentum
hidden	10	0.2	0.3
output	🟥	0.15	0.3

Acting on Δw^{ih} and Δw^{bh}

Acting on Δw^{ho} and Δw^{bo}

Interface 2: Training settings

#	Setting	Value
1	Max. epoch number	200
2	Training/Validation division [%]	85.00%
3	Epoch interval to mix training set	1

Fig. 39: Interface tables of the implemented ANN.

The first interface table (interface 1, cf. Fig. 39) enables to test different configurations of ANN, by changing the number of neurons in the hidden layer and the mathematical hyperparameters which act on weights' corrections (cf. Eq. (10) and Table 34). The number of neurons in the input layer and in the output layer is automatically set by the program, because it depends on the dataset to be processed. The second interface table (interface 2, cf. Fig. 39) involves three important settings impacting on the training process:

- The first setting regards the maximum number of epochs to be performed. When the set number of epochs is achieved, the training process is stopped. At any moment, the training phase can be restarted from the beginning or from the last epoch achieved.
- The second setting enables to automatically generate the validation set (cf. Table 35, fourth guide principle) from the training set. For example, the value of 90% reported in Fig. 39 indicates that the 10% of training samples is randomly selected at the beginning of the training phase, and then used

as validation set during the training process. According to [47,50], this setting is typically limited in the range [70;90]%.

- The third setting regards the epoch interval to mix the order of the training samples. According to the third guide principle in Table 35, this parameter is almost always set to 1, as in Fig. 39.

The implemented ANN is thus tested in an original classification problem regarding image recognition. The considered training set is represented in Fig. 40a, and it consists of 900 samples. The training inputs are images of 15x20 pixels (white and black) depicting numbers from 0 to 9. These images were realized by hand with the program Paint by seven different people, to assure that all the designed numbers are sufficiently different from each other (cf. Table 35, second guide principle). The corresponding training outputs are given in form of probability arrays of 10 elements, where the value 1 identifies the 100% probability of being the number depicted in the image (cf. Fig. 40a). The idea is to teach the ANN to recognize the number depicted in the input image, studying the influence of the number of hidden neurons. Four different numbers of hidden neurons will be tested: 5, 10, 20 and 30. For each one, three different training tests will be conducted to verify the reproducibility of the obtainable training results (and thus the robustness of the implemented ANN[13]), for a total of twelve training tests. The list of the conducted tests is reported in Table 36.

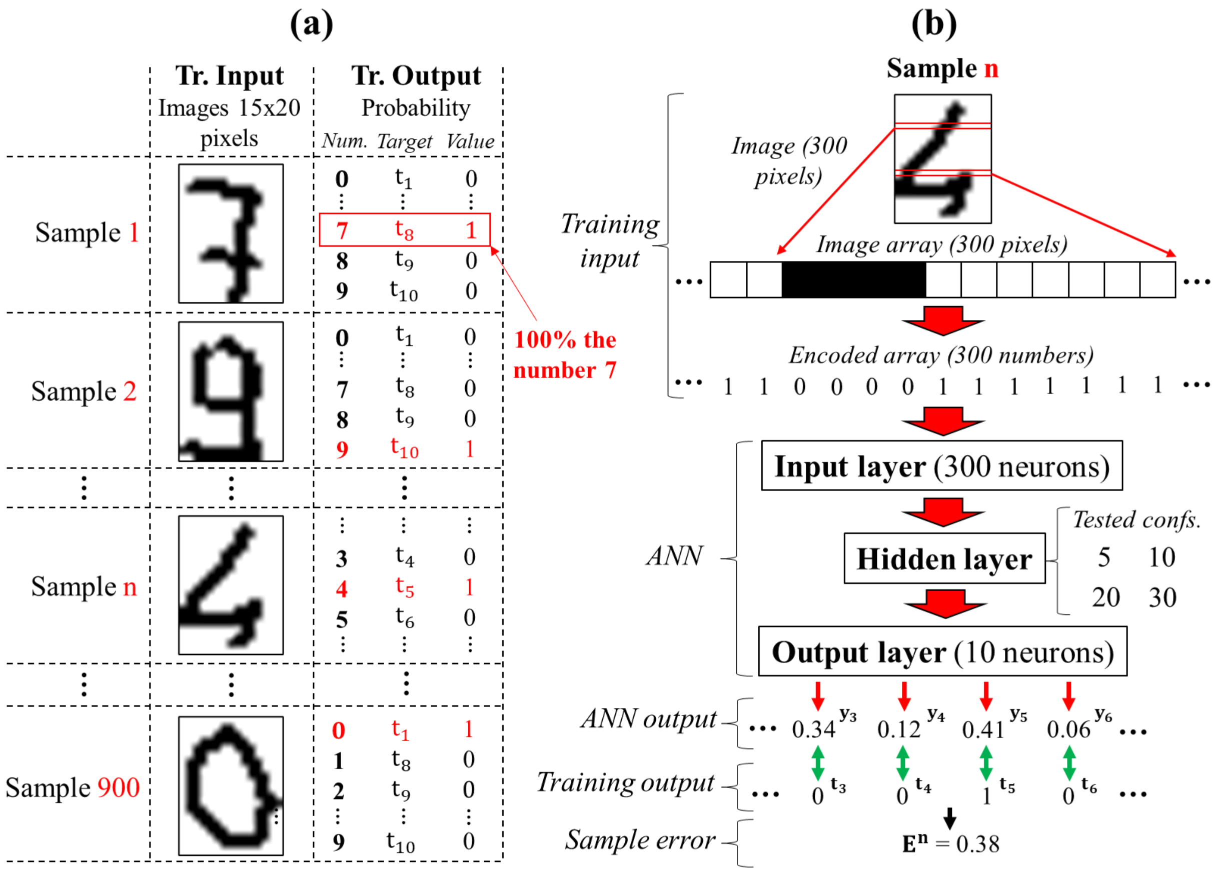


Fig. 40: (a) Original training set for image recognition; (b) forward step of the sample n (cf. Fig. 37b).

[13] Synaptic weights are initialized with random rational numbers at the beginning of the training phase (cf. Fig. 37a). Therefore, all training tests will start with different synaptic weights. The robustness condition is the following: for each tested number of hidden neurons, the three training tests must achieve similar values of E_{ep}^{v} (i.e. a similar accuracy, cf. Table 35).

Table 36: List of the conducted training tests.

Training test #	1	2	3	4	5	6	7	8	9	10	11	12
Number of hidden neurons	5			10			20			30		

Each training test involves two distinct steps:

- Set-up, i.e. the initialization of the ANN configuration and of the training settings in the program interfaces (cf. Fig. 39) before starting the training phase.
- Training, i.e. the "true" training phase (cf. Fig. 29a), which follows the iterative procedure reported in Fig. 37a.

Each step is described in detail as follows.

Set-up. Regarding interface 1, the values of learning rate and momentum are set as in Fig. 39 according to [50,97] and on priorly conducted training tests, while the number of hidden neurons is changed every three tests (cf. Table 36). Regarding interface 2, the training settings have been set as in Fig. 39. This means we will have 810 training samples (90% of the original 900 samples) and 90 validation samples (10% of 900), the latter randomly selected from the original training set (cf. Fig. 40a). The idea is to have as many training samples as possible[14] to allow the ANN to acquire a good classification accuracy, even if the current amount of training samples (810) is lower than the typically suggested minimum (cf. Table 35, first guide principle).

Training. The training phase follows the iterative procedure reported in Fig. 37a. The forward step of the nth training samples is shown in Fig. 40b. Image rows are firstly organized in a pixel array, and then encoded in a binary numerical array, which can be thus given to the ANN. White pixels are converted into 1 and black pixels are converted into 0[15]. As done in Fig. 37b, ANN outputs (y) and training outputs (t) are used to calculate the sample error $\mathrm{E^n}$ by means of Eq. (7). According to Fig. 37a, the following backward step involves the calculation of weights' corrections with the BPA (cf. Eq. (8) and Fig. 38) and their application to synaptic weights by means of Eq. (10). All these operations are repeated for each training samples until the epoch is concluded. At the end of each epoch, the epoch errors $\mathrm{E_{ep}^{t}}$ and $\mathrm{E_{ep}^{v}}$ (cf. Table 35, fourth guide principle) are respectively calculated with Eq. (11) and Eq. (12), which are adapted from Eq. (9).

$$\mathrm{E_{ep}^{t}} = 1 - \frac{\mathrm{N^{R_t}}}{810}, \tag{11}$$

$$\mathrm{E_{ep}^{v}} = 1 - \frac{\mathrm{N^{R_v}}}{90}. \tag{12}$$

In these equations, $\mathrm{N^{R_t}}$ and $\mathrm{N^{R_v}}$ are the numbers of right responses (cf. section 2.2.2.3 and Eq. (9)) provided by the ANN on the training set (810 samples) and on the validation set (90 samples), respectively. New epochs are thus performed until the maximum number of epochs is achieved (i.e. 200, Fig. 39).

[14] 90% is the maximum of the typical range [70;90]% suggested for this training setting [47,50].

[15] This is the most classical method to transform black and white images into numerical inputs.

Let us analyse and discuss the results of the conducted training tests (cf. Table 36). The respective trends of $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ are reported in Fig. 41, Fig. 42, Fig. 43 and Fig. 44, and are all characterised by several epoch ranges where $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ remain constant (plateaux) and by sudden error diminutions.

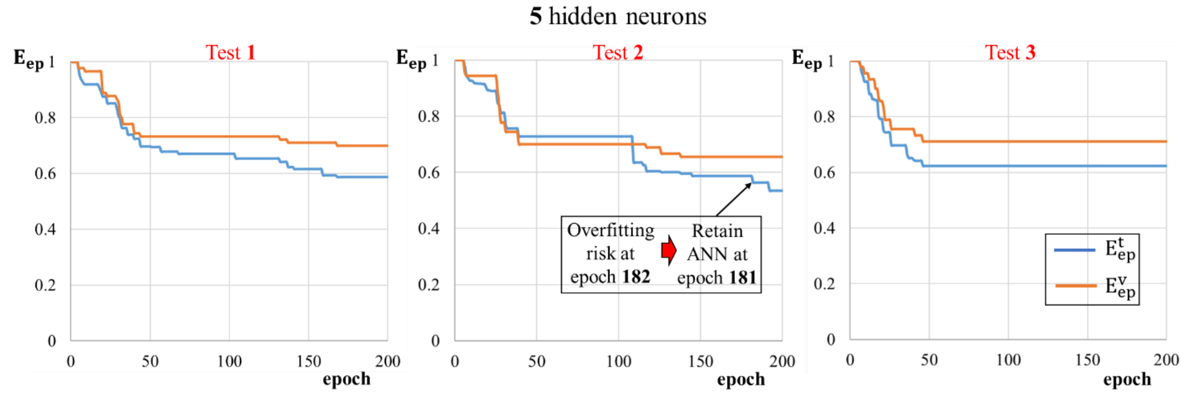


Fig. 41: Trends of E_{ep}^{t} and E_{ep}^{v} in the training tests 1, 2 and 3 (5 hidden neurons, cf. Table 36).

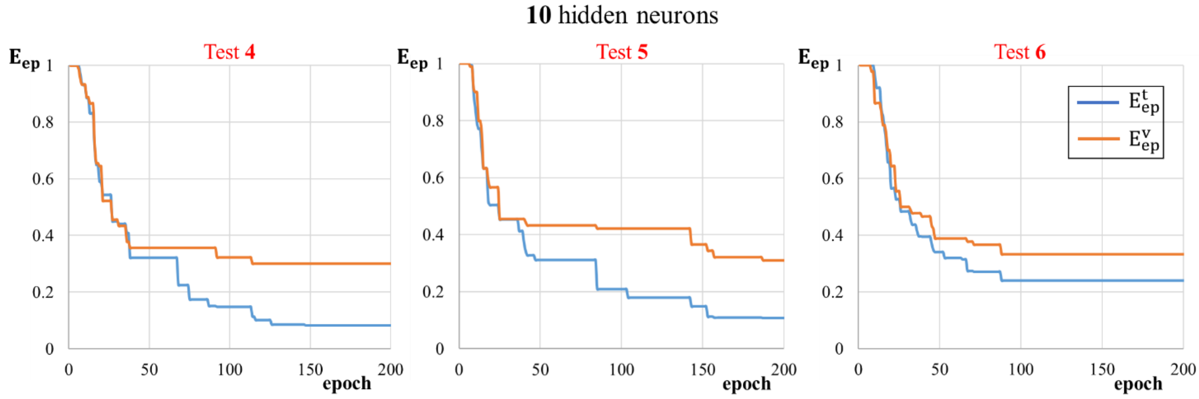


Fig. 42: Trends of E_{ep}^{t} and E_{ep}^{v} in the training tests 4, 5 and 6 (10 hidden neurons, cf. Table 36).

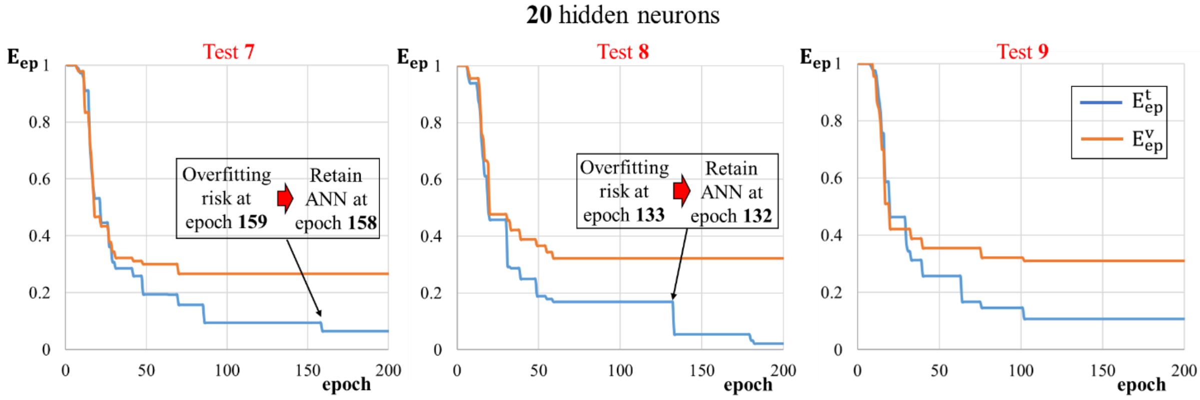


Fig. 43: Trends of E_{ep}^{t} and E_{ep}^{v} in the training tests 7, 8 and 9 (20 hidden neurons, cf. Table 36).

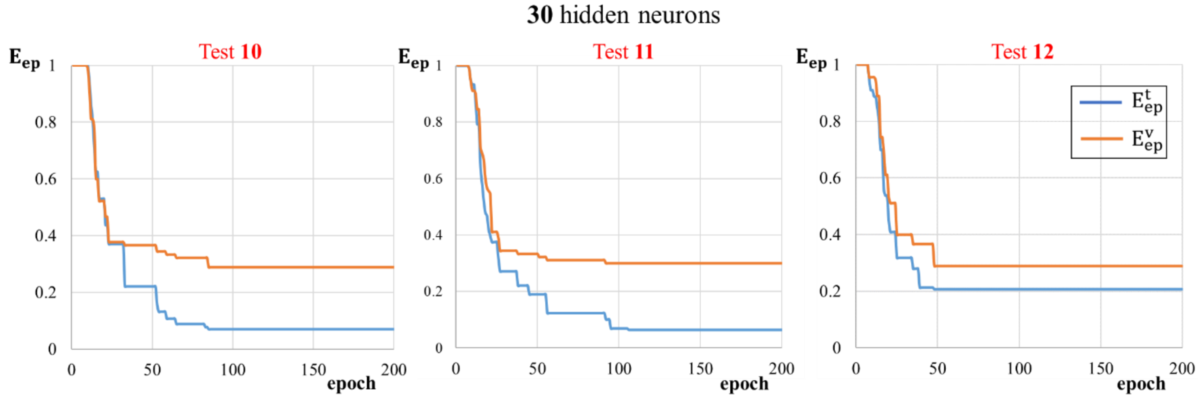


Fig. 44: Trends of E_{ep}^{t} and E_{ep}^{v} in the training tests 10, 11 and 12 (30 hidden neurons, cf. Table 36).

In all the conducted tests, the greatest decrease of $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ is localised in the first 50 epochs. Further error diminutions generally occur up to the 150th epoch. After the 25th-30th epoch, on average, $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ tends to decrease more than $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$. This occurs when few training samples are used (less than a thousand, cf. Table 35), as in this case. According to Table 35 (fourth guide principle), the training samples represent the ensemble of knowledge on which the synaptic weights are directly calibrated, while the role of validation samples is to test the generalization ability acquired by the ANN at the end of each epoch, without participating to weights' corrections (cf. Table 35, fourth guide principle). For this reason, $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ is typically considered more reliable than $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ for the evaluation of the classification accuracy achieved by the trained ANN [47,50]. Monitoring $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ is still very useful to understand if the training process is affected by overfitting (cf. Table 35). Indeed, the reductions of $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ normally occur in the same epoch. However, a potential risk of overfitting exists when $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ continues decreasing while $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ remains constant. This trend can be remarked for tests 2, 7 and 8. In these cases, the training process should be stopped, and the trained ANN at the epoch before overfitting should be retained (cf. Fig. 41 and Fig. 43). The values of $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ achieved during the training tests are summarised in Table 37, together with the average values ($\mathrm{E}_{\mathrm{ep,av}}$), coefficients of variation (COV) and percentage error variations ($\Delta\mathrm{E}_{\mathrm{ep,av}}$) calculated for each tested number of hidden neurons. The red values refer to the tests potentially affected by overfitting. For these tests, the values of $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ at the epoch previous to overfitting (cf. Fig. 41 and Fig. 43) are considered. All the other values refer to the 200th epoch (i.e. the maximum allowed, cf. Fig. 39).

Table 37: Values of E_{ep}^{t} and E_{ep}^{v} achieved in the training tests, and average values ($E_{ep,av}$), variation coefficients (COV) and percentage error variation ($\Delta E_{ep,av}$) for each tested number of hidden neurons.

			Training				*Validation*			
Number hidden neurons	**Test #**	**Epoch #**	$\mathbf{E}_{\mathbf{ep}}^{\mathbf{t}}$	$\mathbf{E}_{\mathbf{ep,av}}^{\mathbf{t}}$	**COV$^{\mathbf{t}}$ [%]**	**$\Delta\mathbf{E}_{\mathbf{ep,av}}^{\mathbf{t}}$ [%]**	$\mathbf{E}_{\mathbf{ep}}^{\mathbf{v}}$	$\mathbf{E}_{\mathbf{ep,av}}^{\mathbf{v}}$	**COV$^{\mathbf{v}}$ [%]**	**$\Delta\mathbf{E}_{\mathbf{ep,av}}^{\mathbf{v}}$ [%]**
5	1	200	0.59	0.60	2	---	0.70	0.69	3	---
	2	181	0.59				0.66			
	3	200	0.62				0.71			
10	4	200	0.08	0.14	48	-76	0.30	0.31	4	-55
	5	200	0.11				0.31			
	6	200	0.24				0.33			
20	7	158	0.10	0.13	24	-7	0.27	0.30	7	-3
	8	132	0.17				0.32			
	9	200	0.11				0.31			
30	10	200	0.07	0.11	60	-15	0.29	0.29	3	-3
	11	200	0.06				0.30			
	12	200	0.21				0.28			

Two main considerations can be made according to Table 37:

- **About the influence of the hidden neurons' number**. For both $\mathrm{E}_{\mathrm{ep}}^{\mathrm{t}}$ and $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$, an important decrease of the average error (-76% and -55%, respectively) can be remarked when the number of hidden neurons is increased from 5 to 10. Much lower error diminutions can moreover be remarked when the number of hidden neurons is further increased. Considering $\mathrm{E}_{\mathrm{ep}}^{\mathrm{v}}$ as a more reliable accuracy indicator, the obtained results suggest that a number of hidden neurons between 10 and 20 is suitable for the current classification problem, with an achieved accuracy around the 70% ($\mathrm{E}_{\mathrm{ep,av}}^{\mathrm{v}} \approx 0.3$). In general, even higher accuracy values can be obtained with ANNs. We however believe the results obtained in the conducted training tests strongly depend on the number of used training samples (810), which is lower than the minimum suggested (cf. Table 35, first guide principle).

- **About the robustness of the implemented ANN**. Based on $\mathrm{COV^t}$ and $\mathrm{COV^v}$, $\mathrm{E_{ep}^t}$ values are generally characterised by a high dispersion, while a much lower dispersion can be remarked for $\mathrm{E_{ep}^v}$. According to [47,50], more interest should be however addressed to $\mathrm{E_{ep}^v}$, since it represents the main parameter for the evaluation of the obtained results. Consequently, the low values of $\mathrm{COV^v}$ suggest the training results are reproducible, thus confirming the robustness of the implemented ANN[13].

It is finally important to remark that the main target of the proposed tests was not to obtain a super accurate ANN to be used in the future, but rather to show how training should be conducted, what type of results are obtained and how they should be interpreted. We believe that the discussed aspects are enough for the comprehension of ANNs and their training process in the context of the current work.

2.2.3. The implemented unsupervised learning-based method: K-means clustering by genetic algorithm (GA)

The implemented unsupervised learning-based method (cf. Table 26) is presented and discussed in this section. As shown in Table 27, any set of samples can be described by means of different features abstracted from the same samples. Based on the definition of abstraction (cf. Table 23), each feature can be isolated and used as CF (clustering feature) to cluster the samples in different ways, as done for example in Fig. 30b. In this section, we want to describe the used clustering method and the algorithm by which it is conducted, highlighting the main parameters and their influence. The current section is organized as follows. An overview and comparison of the two principal clustering approaches, partitional and hierarchical, is firstly proposed in section 2.2.3.1. This enables to better introduce the terminology used in the following sections and to better delimit the adopted clustering method, which is based on partitional clustering. In sections 2.2.3.2 and 2.2.3.3, the adopted partitional clustering method, called K-means, is introduced and illustrated with an example. The genetic algorithm (GA) represents the computational procedure by which the K-means method will be conducted. The base ideas and the procedure flowchart of the GA are presented in section 2.2.3.4, together with an example. The main parameters of the GA are discussed in section 2.2.3.5. The implemented GA is finally presented and tested in 2.2.3.6. The experts of K-means clustering and GA can directly skip to section 2.2.3.6.

2.2.3.1. Overview on the principal clustering approaches: partitional clustering and hierarchical clustering

According to [101–103], two principal clustering approaches exist: partitional clustering and hierarchical clustering. In partitional clustering, the training set is partitioned into various clusters, which are subsequently evaluated by means of some criteria to verify their pertinence (cf. Table 28). An example of partitional clustering is reported in Fig. 30b, where two different partitioning levels[16] (i.e. the two clustering examples) of the same training set can be distinguished. The first partitioning level is based on feature 1 (hair length, cf. Table 27) and includes two different clusters (“long” and “short”, cf. Fig. 30b), while the second partitioning level is based on feature 2 (predominant hair colour, cf. Table 27) and involves three different clusters (“white”, “black” and “brown”, cf. Fig. 30b). There is no hierarchy between these partitioning levels, which are independent from one another [47,102]. In

[16] The term “partitioning level” (or “partitioning”) identifies an ensemble of clusters obtained with the same CFs.

hierarchical clustering, training set elements are combined into clusters, the latter are further partitioned into smaller clusters and so on, creating a hierarchy of partitioning levels. This approach can be considered as an extension of partitioning clustering. Two examples of hierarchical clustering are reported in Fig. 45, where the same training set and clustering features as Fig. 30b are assumed (cf. Table 27).

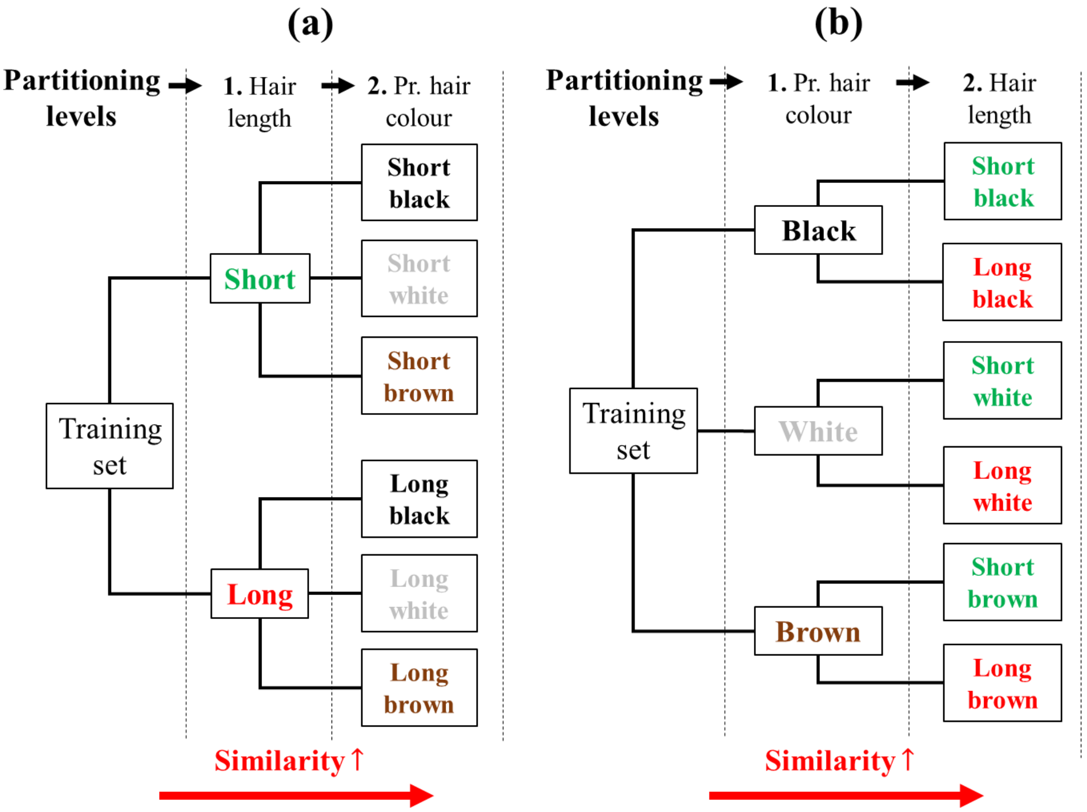


Fig. 45: Based on the example in Fig. 30b: (a) hierarchical clustering according to feature 1 (hair length, cf. Table 27) and feature 2 (predominant hair colour, cf. Table 27); (b) hierarchical clustering according to feature 2 and feature 1.

Contrary to partitional clustering, the partitioning levels are now dependent from one another. In Fig. 45a, the first partitioning level is based on feature 1 (hair length, cf. Table 27), and the second partitioning level is based on feature 2 (predominant hair colour, cf. Table 27). Vice versa in Fig. 45b. The same clusters are obtained in the last partitioning levels. The order of the partitioning levels is arbitrarily decided by user. In both cases, the similarity between the elements of each cluster increases from the left side to the right side of the hierarchical structures (cf. Fig. 45). As for partitioning clustering, the pertinence of the identified clusters must be evaluated a posteriori [101,103]. The two principal clustering approaches are summarised and compared in Table 38.

Table 38: Description and comparison of the two principal clustering approaches.

		Comparison aspects	
Clustering approach	Description	Representation of the generated clusters	Size of the training set
Partitional	The training set is partitioned into various clusters. Different partitioning levels can be constructed based on different CFs (cf. Fig. 30b) and are independent from one another. The pertinence of identified clusters must be evaluated a posteriori.	One different representation for each partitioning level constructed.	Suitable for small and large dataset.
Hierarchical	Training samples are combined into clusters, which are further partitioned into sub-clusters and so forth, creating a hierarchy of partitioning levels, which are thus dependent from one another (cf. Fig. 45). The pertinence of identified clusters must be evaluated a posteriori.	It involves more partitioning levels at once, but it depends on their hierarchical order.	More suitable for small dataset. It requires a more elevated computational effort for large dataset.

Compared to partitional clustering, hierarchical clustering generally provides a more complete representation of data structure which involves more partitioning levels at once. However, the interpretation of the training set is linked to the hierarchical order of the partitioning levels decided by user. Moreover, hierarchical clustering is less suitable than partitional clustering for large dataset, for which the construction of the hierarchical structure generally requires an elevated computational effort [47,101–103]. The clustering method implemented in this work is based on the partitional clustering approach.

2.2.3.2. Introduction to partitional clustering: procedure flowchart and K-means method

The base idea of partitional clustering methods is to determine the minimum number of clusters, so as the dispersion of the training samples in the single clusters is minimum. The general flowchart of a partitional clustering procedure is reported in Fig. 46a (cf. [47,104,105]), and it is characterised by two nested iterative minimization cycles. The external cycle, whose steps are indicated with "e", consists of determining the minimum number of clusters (K) of the final training set partitioning[16]. The internal cycle, whose steps are indicated with "i", corresponds to the step 1e. As shown in Fig. 46a, the internal cycle involves the construction of the training set partitioning (P), composed of K clusters, and the minimization of the associated dispersion (D). The superscript "K" designates the partitioning P of minimum dispersion which results from step 1e (P^K) and the associated dispersion (D^K).

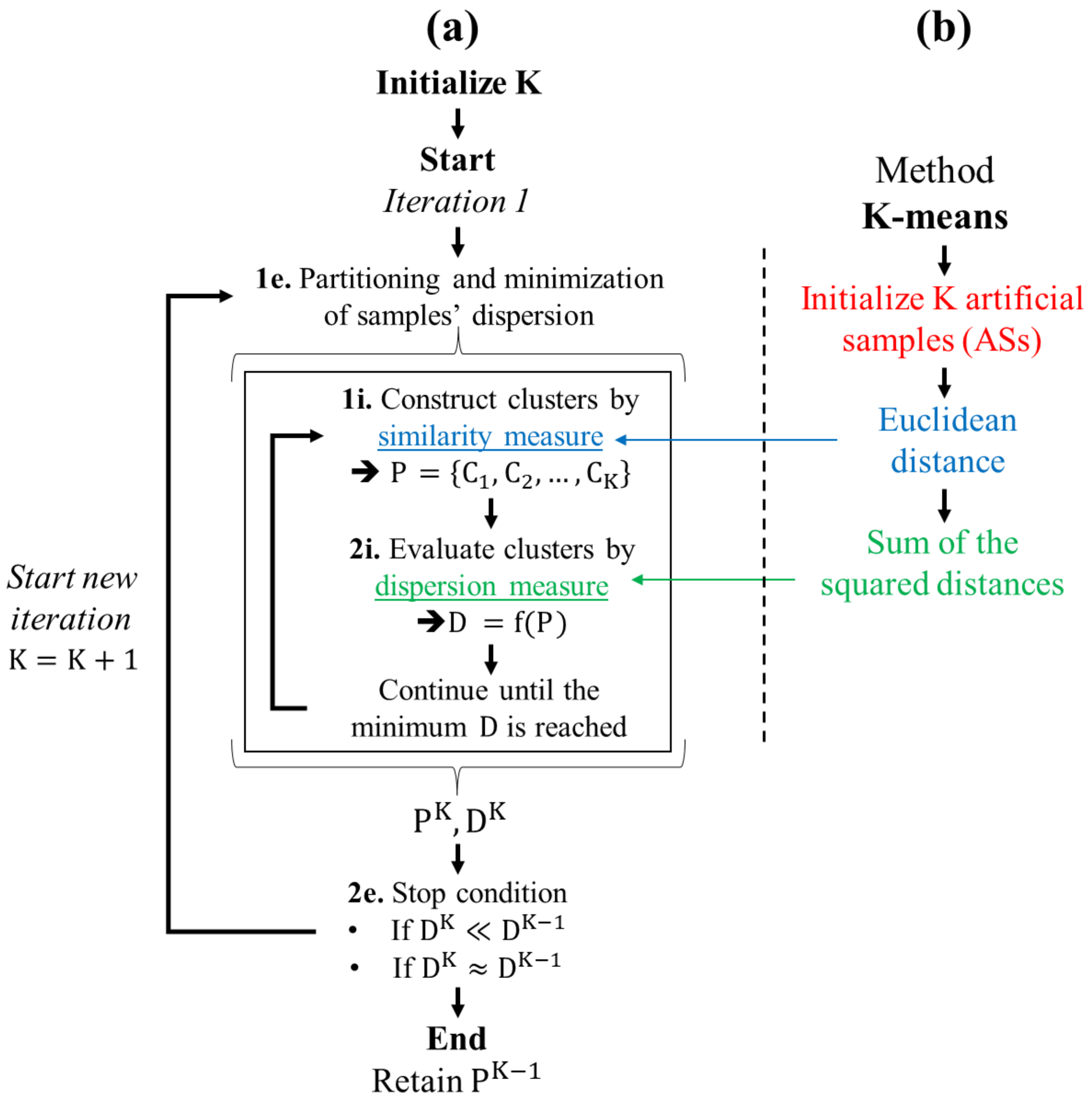

(a)
Initialize K
Start
Iteration 1
1e. Partitioning and minimization of samples' dispersion
1i. Construct clusters by similarity measure
➔ $P = \{C_1, C_2, \dots, C_K\}$
2i. Evaluate clusters by dispersion measure
➔ $D = f(P)$
Continue until the minimum D is reached
P^K, D^K
2e. Stop condition
• If $D^K \ll D^{K-1}$
• If $D^K \approx D^{K-1}$
End
Retain P^{K-1}
Start new iteration
$K = K + 1$
(b)
Method
K-means
Initialize K artificial samples (ASs)
Euclidean distance
Sum of the squared distances

Fig. 46: (a) General flowchart of a partitional clustering procedure; (b) Operations corresponding to the method K-means.

Let us focus on the internal cycle (step 1e). Steps 1i and 2i (cf. Fig. 46a) correspond to the main steps of the partitional clustering approach, i.e. training set partitioning and evaluation of the constructed clusters (cf. Table 38), and are respectively based on the operations of similarity measure and dispersion measure. The differences between the existing methods of partitional clustering mainly reside in how these steps are performed. The similarity measure generally requires the initialization of fixed points in the space of training samples which are used as references to construct the clusters. The K-means method, adopted in this work, consists of initializing these reference points as artificial samples (denoted ASs), i.e. artificially created in the value range of the training samples [104,105]. The measures of similarity and dispersion typically used in the K-means method are reported in Fig. 46b. As argued in [106], K-means is one of the most widely used clustering methods, and its success is mainly due to easy application to many practical cases. The clustering procedure and the K-means method are illustrated with the following example. Let us consider the training set reported in Table 39, composed of 10 different training samples defined in R^2. According to Table 27, the coordinates x and y, also indicated as Si_1 and Si_2, represent the features of the considered training samples. Let us start the clustering procedure with K=2: in the first iteration of the external cycle (cf. Fig. 46a), the training set must be partitioned into two clusters. According to the K-means method (cf. Fig. 46b), we must designate two ASs (one for each cluster to construct). The designated ASs are reported in Table 40, where the values of the features ASj_1 and ASj_2 have been arbitrarily initialized in the value range of Si_1 and Si_2.

Table 39: Considered training set.

Sample (Si)	Coordinate x (Si_1)	Coordinate y (Si_2)
S1	2	2
S2	1	3
S3	4	7
S4	3	4
S5	9	4
S6	7	6
S7	2	9
S8	6	8
S9	7	3
S10	8	7

Table 40: Designated ASs.

Sample (ASk)	Coordinate x (ASj_1)	Coordinate y (ASj_2)
AS1	4	3
AS2	6	7

With the help of Fig. 46a, let us see in detail how steps 1i and 2i are performed.

Step 1i. According to the K-means method, the clusters are constructed by measuring the similarity between the designated ASs and the training samples. The similarity measure is performed with Eq. (13):

$$\mathrm{d(ASj, Si)} = \sqrt{\sum_{h=1}^{n} (\mathrm{ASj_h} - \mathrm{Si_h})^2}\,, \tag{13}$$

i.e. the Euclidean (cf. Fig. 46b) distance between the generic artificial sample ASj = (ASj_1, ASj_2, …, ASj_n) and the generic training sample Si = (Si_1, Si_2, …, Si_n), where n is the number of dimensions (i.e. the number of features) of the considered training set. The lower d(ASj, Si), the more similar ASj and Si. The distances calculated with Eq. (13) between the designated ASs (cf. Table 40) and the considered training samples (cf. Table 39) are reported in Table 41. In this case, both x and y have been adopted as CFs, since both features (Si_1 and Si_2) are involved in the similarity measure.

Table 41: Euclidean distances between the designated ASs and the considered training samples, and corresponding clusters.

Sample (Si)	d(AS1,Si)	d(AS2,Si)	Lower AS	Cluster
S1	2.24	6.4	AS1	C_1
S2	3	6.4	AS1	C_1
S3	4	2	AS2	C_2
S4	1.41	4.24	AS1	C_1
S5	5.1	4.24	AS2	C_2
S6	4.24	1.41	AS2	C_2
S7	6.32	4.47	AS2	C_2
S8	5.39	1	AS2	C_2
S9	3	4.12	AS1	C_1
S10	5.66	2	AS2	C_2

According to the calculated values, the samples S1, S2, S4 and S9 are more similar to the AS1, while the samples S3, S5, S7, S8 and S10 are more similar to AS2. The actual partitioning P is thus characterised by the clusters C_1 = {S1, S2, S4, S9} and C_2 = {S3, S5, S6, S7, S8, S10} (cf. Table 41). C_1 and C_2 are represented in Fig. 47a, where the round ticks correspond to the training samples (cf. Table 39) and the "X" ticks to the designated ASs (cf. Table 40). The content of C_1 and C_2 can be varied by changing the adopted CFs. Adopting CF = x involves that only x is considered in the similarity measure, i.e. in Eq. (13). The resulting clusters C_1 = {S1, S2, S3, S4, S7} and C_2 = {S5, S6, S8, S9, S10} are represented in Fig. 47b. Otherwise, adopting CF = y involves that only y is considered in Eq. (13). The obtained clusters C_1 = {S1, S2, S4, S5, S9} and C_2 = {S3, S6, S7, S8, S10} are represented in Fig. 47c.

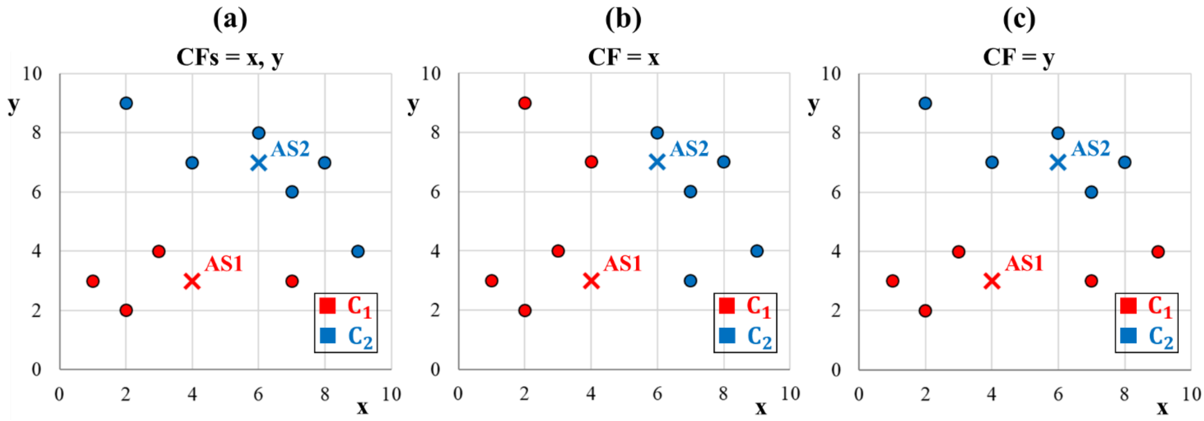


Fig. 47: Representation of the clusters C_1 and C_2 obtained by adopting different CFs: (a) CFs = x, y; (b) CF = x; (c) CF = y.

Step 2i. Based on the adopted similarity measure (cf. Eq. (13)), the dispersion measure will describe how much the samples contained in each cluster are distant from the designated ASs. As indicated in Fig. 46b, the dispersion associated to the partitioning P is measured as the sum of the squared distances between the clustered samples and the respective ASs, and it is expressed by Eq. (14),

$$D = \sum_{j=1}^{K} D_j = \sum_{j=1}^{K} \sum_{i=1}^{N_j^S} d(ASj, Si(j))^2 \, , \qquad (14)$$

where:

- K is the number of clusters in the actual partitioning P = {C_1, C_2, …, C_K}.
- D_j is the dispersion contribution due to the jth cluster.
- N_j^S is the number of training samples contained in the jth cluster.
- ASj is the artificial sample (i.e. the reference) of the jth cluster.
- Si(j) is the generic training sample contained in the jth cluster.

The lower D, the better the overall partitioning. Moreover, due to square, the more training samples are distant from the respective AS, the more weight they have on the dispersion D [104,105]. For example, let us consider the partitioning P = {C_1, C_2} obtained in Table 41 and represented in Fig. 47a. Eq. (14) leads to D = 74. The weight of the single clusters and of single samples on D can be observed in Table 42 (cf. Eq. (14)).

Table 42: Weight of clusters and training samples on the partitioning dispersion (cf. Table 41).

Cluster	ASj	Si	$d(ASj,Si)^2$	$\frac{d(ASj,Si(j))^2}{D}$ [%]	D_j	$\frac{D_j}{D}$ [%]
C_1	AS1	S1	5	6.8	25	33.8
		S2	9	12.2		
		S4	2	2.7		
		S9	9	12.2		
C_2	AS2	S3	4	5.4	49	66.2
		S5	18	24.3		
		S6	2	2.7		
		S7	20	27		
		S8	1	1.4		
		S10	4	5.4		

According to Table 42, the highest contribution to D is given by the cluster C_2 (66.2 %), and notably by the samples S5 (24.3 %) and S7 (27 %), which are the most distant from AS2 (cf. Table 41).

2.2.3.3. Dispersion minimization in the internal cycle and choice of the final partitioning

Based on Fig. 46a, the internal cycle consists of continually varying the position of the ASs until the minimum dispersion of the constructed partitioning is reached. Let us resume the previous example, where the ASs designated in Table 41 led to the global dispersion D = 74 (cf. Table 42). According to the K-means method (cf. Fig. 46b), let us re-initialize the two ASs with different values of x and y and construct a new partitioning by performing the steps 1i and 2i (cf. Fig. 46a) with Eqs. (13) and (14). The partitioning P = {C_1, C_2} represented in Fig. 48a is obtained with AS1 = (2.4, 5) and AS2 = (7.4, 5.6), and is characterised by D = 63 (<74). In general, the internal cycle is continually re-started with different ASs until D converges towards a minimum value. Notably, the minimization of D represents the most challenging aspect regarding the K-means method [104–106], and it will be better discussed in section 2.2.3.4. In this case, the partitioning in Fig. 48a corresponds to the partitioning with the minimum dispersion obtainable for K=2.

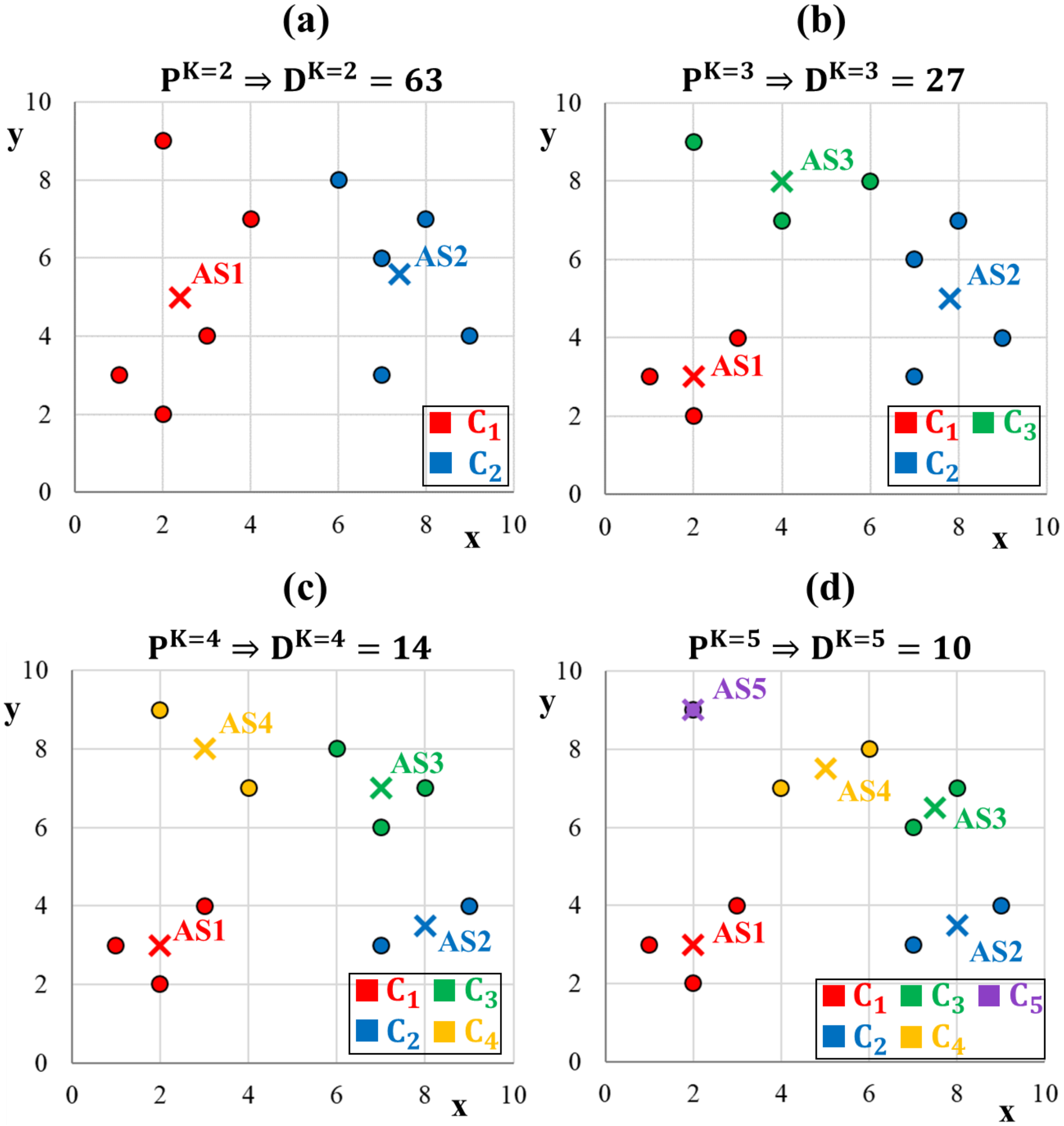


Fig. 48: Partitioning levels with minimum dispersion for (a) K=2, (b) K=3, (c) K=4 and (d) K=5.

According to Fig. 46a, step 1e is thus concluded, and the resulting P^K and $D^K = 63$ (cf. Fig. 48a) are passed to step 2e. As shown in Fig. 46a, the stop condition in step 2e requires comparing D^K to the minimum dispersion obtained in the previous iteration, i.e. D^{K-1}. In this case, step 1e has been directly performed for K=3, K=4 and K=5 to study the influence of K on D^K. The resulting partitioning levels are respectively shown in Fig. 48b, c and d. The samples contained in the clusters and the position of the ASs in the partitioning levels of Fig. 48 are summarised in Table 43.

Table 43: Content of the clusters and position of the ASs for $P^{K=2}$, $P^{K=3}$, $P^{K=4}$ and $P^{K=5}$ (cf. Fig. 48a, b, c, d).

				Corresponding ASs		
K	**Partitioning**	**Cluster**	**Samples (Si)**	**ASj**	**x (ASj₁)**	**y (ASj₂)**
2	$P^{K=2}$	C_1	S1, S2, S3, S4, S7	AS1	2.4	5
		C_2	S5, S6, S8, S9, S10	AS2	7.4	5.6
3	$P^{K=3}$	C_1	S1, S2, S4	AS1	2	3
		C_2	S5, S6, S9, S10	AS2	7.8	5
		C_3	S3, S7, S8	AS3	4	8
4	$P^{K=4}$	C_1	S1, S2, S4	AS1	2	3
		C_2	S5, S9	AS2	8	3.5
		C_3	S6, S8, S10	AS3	7	7
		C_4	S3, S7	AS4	3	8
5	$P^{K=5}$	C_1	S1, S2, S4	AS1	2	3
		C_2	S5, S9	AS2	8	3.5
		C_3	S6, S10	AS3	7.5	6.5
		C_4	S3, S8	AS4	5	7.5
		C_5	S7	AS5	2	9

Based on Fig. 48, the ASs reported in Table 43 correspond to the centroids of the respective clusters. Moreover, as K increases, the positions of the ASs tend to coincide with the training samples' ones. For example, the position of AS5 in Fig. 48d coincides with the position of S7 (cf. Table 43). The values of D^K obtained from K=2 to K=5 (cf. Fig. 48a, b, c and d) are summarised in Table 44, where R_D represents the total percentage reduction of D^K due to the increasing of K, and it is expressed by Eq. (15):

$$R_D = 1 - \frac{D^K}{D^{K=1}}, \tag{15}$$

and ΔR_D represents the increment of R_D between two adjacent values of K.

Table 44: Values of D^K and associated R_D (cf. Fig. 48a, b, c, d and Eq. (15)).

K	**D^K**	**R_D [%]**	**ΔR_D [%]**
1	125	0	---
2	63	50	+50
3	27	78	+28
4	14	89	+11
5	10	92	+3

$D^{K=1}$ = 125 (cf. Table 44) represents the dispersion of the training samples (cf. Table 39) around their centroid (x = 4.9, y = 5.3) and it is calculated with Eq. (14). Based on [104–106], $D^{K=1}$ corresponds to the "worst partitioning scenario" (one unique cluster containing all the training samples) and it is thus employed as reference value for calculating R_D (cf. Eq. (15)). The more K increases, the more D^K reduces and the more R_D increases. We can see that D^K tends to significantly reduce for K=2 (ΔR_D = +50%, cf. Table 44) and K=3 (ΔR_D = +28%, cf. Table 44). A less meaningful reduction of D^K can be moreover remarked when K is increased from 3 to 4 (ΔR_D = +11%, cf. Table 44). Much lower dispersion diminutions are obtained for K>4. The curve K-R_D reported in Fig. 49 is finally used to select the final partitioning.

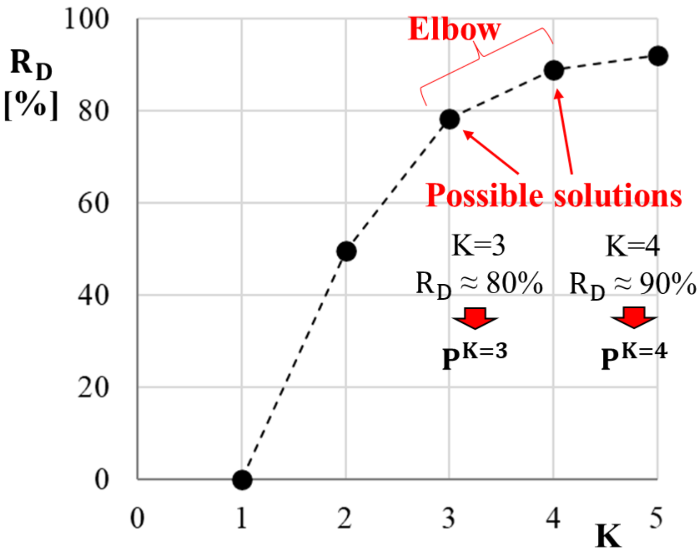


Fig. 49: Curve K-R_D (cf. Table 44).

According to [104–106] and Fig. 46a, the final partitioning solution is a compromise between minimizing K and minimizing D^K. The relative weight of these factors is clearly decided by the user. As shown in Fig. 49, the elbow of the curve K-R_D separates the zone with a high R_D variation (K≤3) from the zone with a weak R_D variation (K≥4), and it thus represents the ensemble of the compromise solutions. In general, the elbow of the curve K-R_D can be more or less accentuated depending on the training set, making the choice of the partitioning solution more or less complicated. As indicated in Fig. 49, the partitioning levels $P^{K=3}$ and $P^{K=4}$ (cf. Fig. 48b, c and Table 43) represent the two possible solutions of the clustering procedure. With $P^{K=3}$, we have less clusters, globally characterised by a higher dispersion. With $P^{K=4}$, we have more clusters, globally characterised by a lower dispersion.

2.2.3.4. The genetic algorithm (GA): base ideas and procedure flowchart

In step 1e (internal cycle, cf. Fig. 46a), the partitioning dispersion (D) is minimized by varying the position of the designated ASs (ASj_1, ASj_2, …, ASj_n, cf. Table 40) in the space of the training samples. Notably, this process requires the implementation of an optimization algorithm able to automatically handle the variation of ASs' position, and it thus represents the most challenging aspect regarding the K-means method. As argued in [104–106], most of the algorithms which are classically used for this task are based on gradient descend (gradient algorithms). This approach is however very sensible to ASs' initial positions, involving a high risk of getting trapped in local minima of D during the optimization procedure. We thus adopted an alternative optimization technique which is much less affected by these problems: the genetic algorithm (GA). The GA is inspired to the Darwinian laws of the natural evolution, from which it takes two base aspects:

- The natural selection, i.e. the process enabling a population of individuals to change over the generations. The individuals endowed with more advantageous survival characteristics have a higher probability to survive and reproduce. Similarly, a GA starts from an initial population of solutions, which is randomly generated at the beginning of the optimization process. At each iteration (generation), the solutions are evaluated, and a score (fitness) is assigned to each one: the higher the fitness, the better the solution. Based on fitness, different solutions are then selected as "parents" and recombined each other (reproduction) into a new population of solutions ("sons"). The parental selection favours the solutions with a higher fitness, which thus tend to transmit their good characteristics to the following generations [107,108].

- The hereditary transmission mechanisms of the genetic characteristics (genes) from a generation to the following ones. To emulate these mechanisms, the solutions are encoded in a binary representation. As shown in Fig. 50a, each solution in the population is represented by a chromosome, i.e. an ensemble of genes characterised by the bits 1 or 0. Cross-over and mutation are the two genetic mechanisms emulated in a GA and are shown in Fig. 50b. During reproduction, the genetic materials of the parents is firstly recombined by cross-over (chromosomic “cut” and “exchange”) and transmitted to the sons, whose genes are afterwards subjected to a mutation (bit flip from 1 to 0 or vice versa). In general, the cross-over cutting points and the mutating genes (cf. Fig. 50b) are randomly selected.

The two base aspects of the GA are summarised in Table 45.

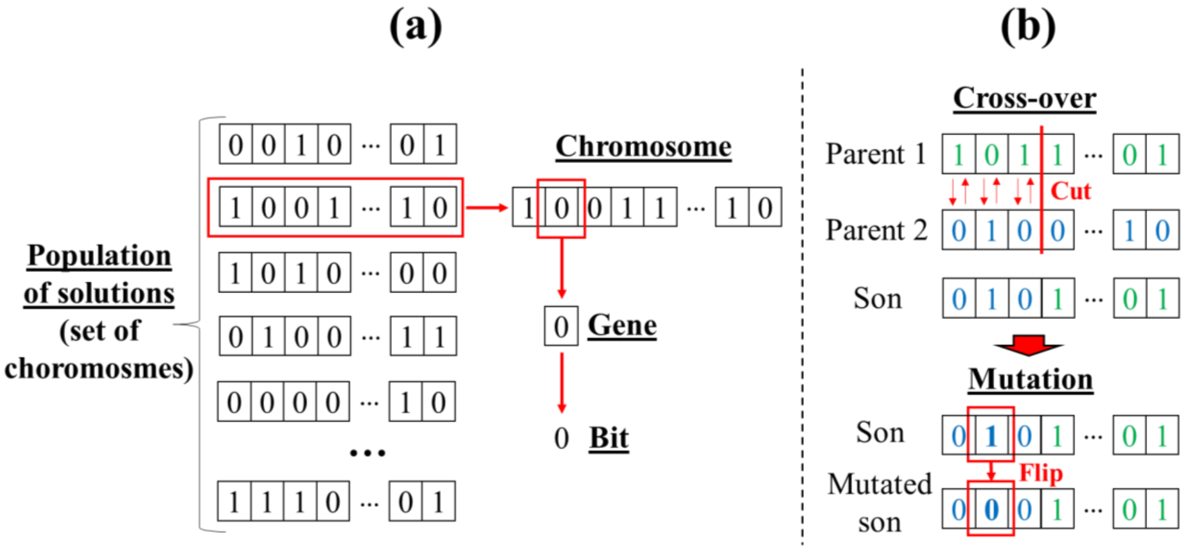


Fig. 50: (a) Binary representation of the solutions in a GA; (b) mechanisms of genetic transmission: cross-over and mutation.

Table 45: Base aspects of the GA inspired to the natural evolution.

Aspect	Description
Natural selection	An initial population of randomly generated solutions is iteratively evolved over several generations. At each generation, the solutions with higher fitness (better and more advantageous characteristics) have a higher probability to be selected as parents of the new generation.
Hereditary transmission mechanisms of the genetic characteristics (genes)	Representation of the solution in binary code (chromosome, cf. Fig. 50a). Emulation of the following genetic mechanisms (cf. Fig. 50b): • Cross-over: recombination of the parents’ genetic material during reproduction, emulated by means of chromosomic “cut” and “exchange”. • Mutation: random change of sons’ genetic characteristics emulated by flipping the bit in the selected mutating gene.

The procedure flowchart associated to the GA is reported in Fig. 51 and it is divided into two parts (a and b): Fig. 51a shows the preliminary steps and Fig. 51b the iterative phase of optimization, where G (generation) indicates the generic iteration. The procedure in Fig. 51 is illustrated by the following example. Let us perform the step 1e of the clustering procedure in Fig. 46a with the GA, by considering the training set in Table 39 and by assuming K=2. Our variables thus correspond to the coordinates x and y of the two ASs to determine, i.e. $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$, whose values will be limited in the range [0;10] (cf. Table 39). Both the features x and y are adopted as CFs. The GA will determine the values of $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$ corresponding to the partitioning of minimum dispersion, i.e. $P^{K=2}$ (cf. Fig. 46a).

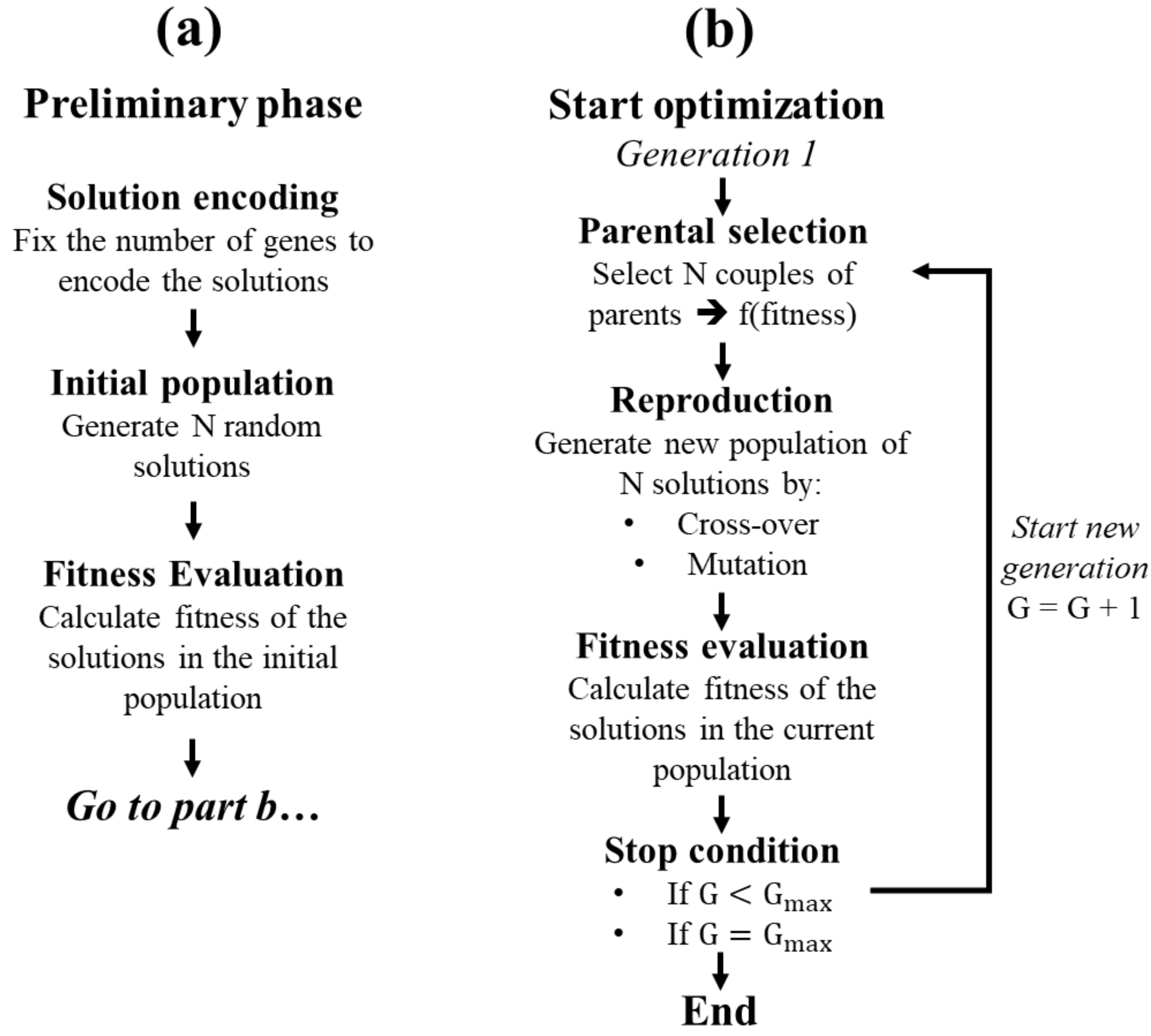

(a)
Preliminary phase
Solution encoding
Fix the number of genes to encode the solutions
Initial population
Generate N random solutions
Fitness Evaluation
Calculate fitness of the solutions in the initial population
Go to part b…
(b)
Start optimization
Generation 1
Parental selection
Select N couples of parents → f(fitness)
Reproduction
Generate new population of N solutions by:
• Cross-over
• Mutation
Fitness evaluation
Calculate fitness of the solutions in the current population
Stop condition
• If G < Gmax
• If G = Gmax
End
Start new generation
G = G + 1

Fig. 51: GA flowchart: (a) preliminary phase and (b) iterative optimization phase.

Each step of the procedure in Fig. 51 is described in detail as follows.

Solution encoding. According to Fig. 51a, this step consists in fixing the number of genes used in the binary representation of the solution (chromosome, cf. Fig. 50a). As shown in Fig. 52, the generic chromosome is characterised by four ordered sections, each one inherent to one of the considered variables and with the same number of genes (g). In each section of the chromosome, the genes represent the value of the corresponding variable expressed in binary code. The possible values that each variable can assume during the optimization procedure depend on g.

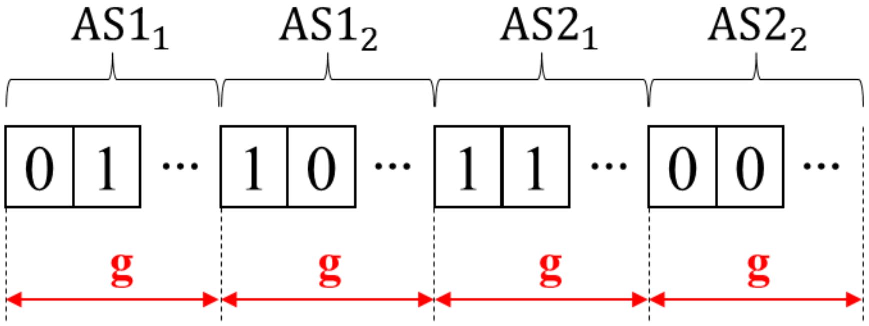


Fig. 52: Structure of the generic chromosome.

Let us adopt the symbol "ASj_h" (cf. Eq. (13)) for generically referring to the variables $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$. Based on Fig. 52, ASj_h has 2^g possible values which are equally spaced in the range [0;10]. The distance increment between two values (Δr) is expressed by Eq. (16):

$$\Delta r = \frac{r_{max} - r_{min}}{(2^g - 1)}, \tag{16}$$

where r_{min} and r_{max} correspond to the range boundaries and are respectively equal to 0 and 10. Let us see some examples:

- If g = 1, ASj_h will have only 2 possible values, i.e. 0 and 10, corresponding to the range boundaries. In the chromosome (cf. Fig. 52), these values will be respectively represented by the bits 0 and 1.
- If g = 2, ASj_h will have $2^2 = 4$ possible values spaced of Δr = 3.33 (cf. Eq. (16)). The possible values and their corresponding binary representations are reported in Table 46.

Table 46: Possible values of the generic variable ASjh and corresponding binary representations for g = 2.

		Binary representation	
#	**Value**	**Gene 1**	**Gene 2**
1	0	0	0
2	3.33	0	1
3	6.67	1	0
4	10	1	1

- If g = 3, ASj_h will have $2^3 = 8$ possible values spaced of Δr = 1.43 (cf. Eq. (16)). The possible values and their corresponding binary representations are reported in Table 47.

Table 47: Possible values of the generic variable ASj_h and corresponding binary representations for g = 3.

		Binary representation		
#	**Value**	**Gene 1**	**Gene 2**	**Gene 3**
1	0	0	0	0
2	1.43	0	0	1
3	2.86	0	1	0
4	4.29	0	1	1
5	5.71	1	0	0
6	7.14	1	0	1
7	8.57	1	1	0
8	10	1	1	1

The higher g, the more refined the optimization space (cf. Table 46 and Table 47), and the more time the optimization procedure will take. For brevity, g = 2 is adopted in the current example. Our chromosomes will be thus composed of four couples of genes corresponding to the variables $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$, in the same order as Fig. 52. Table 46 will be used in the next steps of the procedure (cf. Fig. 51) for decoding the binary representations of the variables.

Initial population. The current step involves the generation of an initial population of N random solutions (cf. Fig. 51a). According to [107,108], the choice of N is typically based on literature indications and on priorly conducted optimization tests. For brevity, N = 5 is adopted in the current example. Our initial population thus consists of 5 different solutions, which are reported in Table 48. The binary representation of each variable can be decoded by means of Table 46. The decoded values are reported in Table 49. Each solution thus represents a specific combination of the variables $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$.

Table 48: Initial population of randomly generated chromosomes.

	$AS1_1$		$AS1_2$		$AS2_1$		$AS2_2$	
Solution #	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**
1	1	1	0	0	0	1	1	1
2	1	1	0	1	0	0	0	1
3	0	0	1	0	1	0	1	1
4	0	1	1	1	0	0	0	1
5	0	0	0	0	0	1	1	1

Table 49: Values corresponding to the encoded variables in Table 48 (cf. Table 46).

Solution #	$AS1_1$	$AS1_2$	$AS2_1$	$AS2_2$
1	10	0	3.33	10
2	10	3.33	0	3.33
3	0	6.67	6.67	10
4	3.33	10	0	3.33
5	0	0	3.33	10

Fitness evaluation. Based on Table 45, the fitness is an indicator of how good a given solution is: the higher the fitness, the better the solution. The fitness of our solutions depends on the dispersion (D) of the partitioning levels constructed with the ASs' coordinates in Table 49. As done in section 2.2.3.2, the partitional clustering (step i1 and i2, cf. Fig. 46a) of the training set (cf. Table 39) is first performed for all solutions in the initial population (cf. Table 49), by using the similarity and the dispersion measures (cf. Eq. (13) and (14)). The fitness of each solution is then calculated with Eq. (17) (cf. [107,108]):

$$\mathrm{fitness} = \frac{\frac{1}{\mathrm{D}}}{\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\frac{1}{\mathrm{D_i}}\right)}, \tag{17}$$

where D is the dispersion of the corresponding partitioning. The lower D, the higher the fitness. The values of D and of fitness associated to the solutions of the initial population are reported in Table 50.

Table 50: Partitioning dispersion (D) and fitness of the solutions (chromosomes) in the initial population.

Solution #	D	Fitness
1	275	0.16
2	164	0.26
3	204	0.21
4	217	0.20
5	253	0.17

Thanks to Eq. (17), the sum of all fitness values is always equal to 1 (cf. Table 50). As suggested in [107,108], this enables to consider the fitness as a percentage probability indicator: the higher the fitness, the more probable the corresponding solution is selected as parent of the new population (cf. Table 45). According to Table 50, the highest fitness belongs to solution 2, characterized by AS1 = (10, 3.33) and AS2 = (0, 3.33) (cf. Table 49), while the lowest fitness belongs to solution 1, characterized by AS1 = (10, 0) and AS2 = (3.33, 10) (cf. Table 49). The preliminary steps of the GA are now concluded, and the first generation of the optimization phase can thus start (cf. Fig. 51).

Parental selection. This step consists of selecting N couples of parents from the initial population which will be used to generate a new population of solutions in the following reproduction step (cf. Fig. 51b). The fitness values reported in Table 50 represent the probability of each solution in the initial population to be selected as parent. The parental selection is performed with the so-called roulette method [107,108]. As shown Fig. 53a, the initial population can be represented in a pie chart, where the angular width of the sections is proportional to the fitness values of the solutions (cf. Table 50). Let us imagine rotating the pie chart like a roulette of a casino for 2N times. At the end of each rotation, the solution indicated by the parent selector is selected as parent. Clearly, the solutions with a higher fitness may be selected more than once, while those with a lower fitness have a lower probability to be selected. As shown in Fig. 53b, the roulette method can be easily implemented by representing the population as an ensemble of probability ranges, whose widths are proportional to the fitness value of the solutions. A random number between 0 and 1 is thus generated by the machine and used to select the parents, as done in Fig. 53b. The couples of selected parents are reported in Table 51.

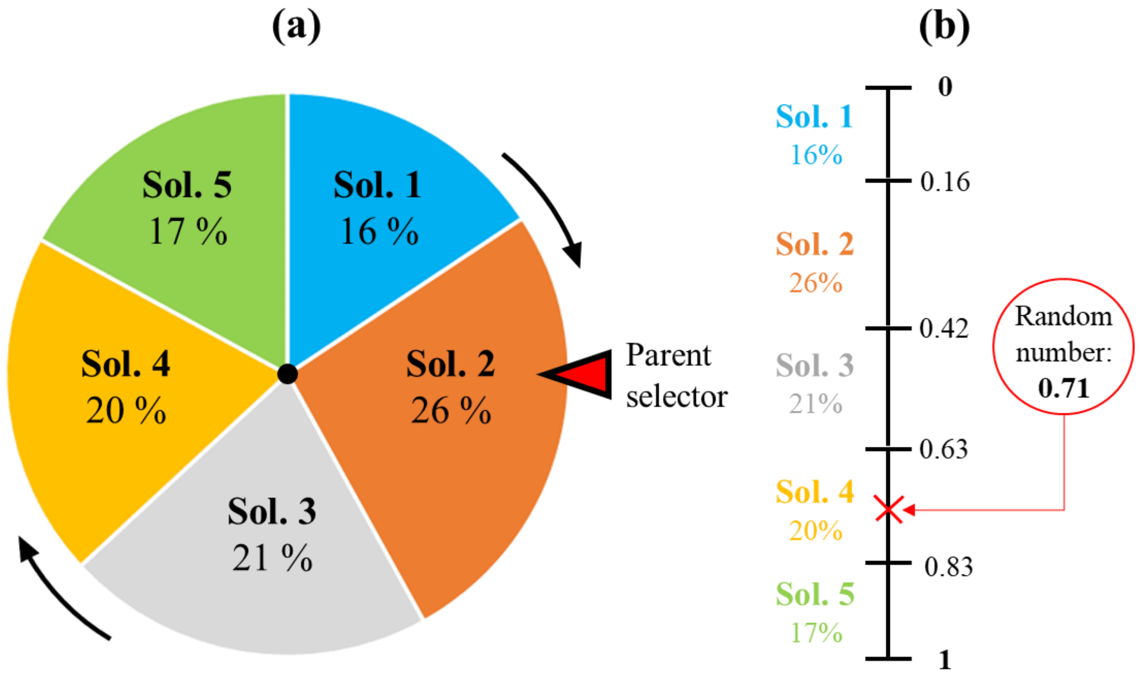


Fig. 53: (a) Roulette method for the parental selection and (b) computational implementation of the roulette method.

Table 51: Couples of selected parents.

	Couple 1	Couple 2	Couple 3	Couple 4	Couple 5
Parent 1 (solution #)	4	1	2	2	3
Parent 2 (solution #)	3	2	3	1	5

In this case, all initial solutions have been selected at least once. Solution 2 and 3 are more frequent due to their high fitness values (26% and 21%, respectively), while the other solutions are less frequent.

Reproduction. A new population of N solutions is generated from the initial population (cf. Fig. 51b). Each couple of parents (cf. Table 51) generates one new solution, by means of the cross-over and mutation mechanisms (cf. Fig. 50b). The employed types of cross-over and mutation are respectively represented in Fig. 54a and b. According to [107,108], the use of both single and double mechanisms enables to recombine the parents' genetic characteristics in more diverse ways, and this increases the probability to generate better solutions. For both single and double mechanisms, the positions of the cross-over cutting points (cf. Fig. 50b) and of the mutating genes is randomly decided just before crossing the parents and are different for each couple of parents.

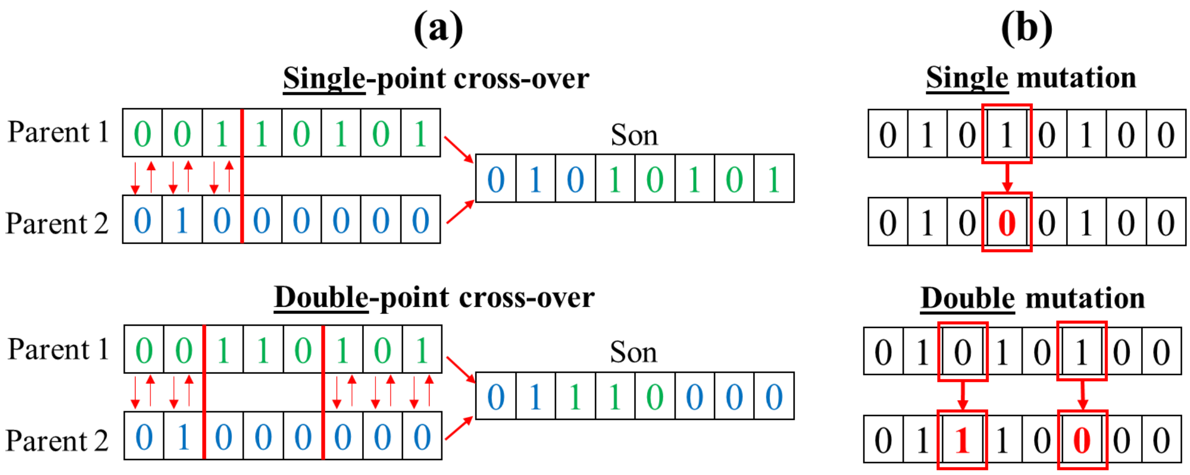


Fig. 54: (a) Used types of cross-over and (b) used types of mutation.

The generation of the 5 new solutions from the 5 selected couples of parents (cf. Table 51) is shown in Table 52. The colours green and blue help to better visualize the performed cross-over mechanisms as done in Fig. 50b and Fig. 54a, while the genes affected by mutation are highlighted in red as in Fig. 54b. Notably, couples 1 and 3 have been crossed with a double-point cross-over, while couples 2, 4 and 5 with a single-point cross-over (cf. Fig. 54a). Moreover, sons 1 and 2 have been subjected to a double mutation, while sons 3, 4 and 5 to a single mutation (cf. Fig. 54b). The chromosomes of the new generated solutions (sons, cf. Table 52) have been decoded by means of Table 46, and the values of the variables have been reported in Table 53.

Table 52: Generation of the new population from the selected parents (cf. Table 51) by cross-over and mutation.

		$AS1_1$		$AS1_2$		$AS2_1$		$AS2_2$	
	Solution #	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**	**Gene 1**	**Gene 2**
Couple 1	4	0	1	1	1	0	0	0	1
	3	0	0	1	0	1	0	1	1
Son 1	1	0	0	1	**0**	**1**	0	1	1
Couple 2	1	1	1	0	0	0	1	1	1
	2	1	1	0	1	0	0	0	1
Son 2	2	1	1	**1**	0	0	**0**	1	1
Couple 3	2	1	1	0	1	0	0	0	1
	3	0	0	1	0	1	0	1	1
Son 3	3	0	1	0	1	1	0	**0**	1
Couple 4	2	1	1	0	1	0	0	0	1
	1	1	1	0	0	0	1	1	1
Son 4	4	**0**	1	0	1	0	0	0	1
Couple 5	3	0	0	1	0	1	0	1	1
	5	0	0	0	0	0	1	1	1
Son 5	5	0	**1**	0	0	0	0	1	1

Table 53: Values of the variables in the new population.

Solution #	$AS1_1$	$AS1_2$	$AS2_1$	$AS2_2$
1	0	6.67	6.67	10
2	10	6.67	0	10
3	3.33	3.33	6.67	3.33
4	3.33	3.33	0	3.33
5	3.33	0	0	10

Fitness evaluation. According to Fig. 51b, let us evaluate the fitness of the solutions in the new population (cf. Table 53). As done in the preliminary phase (cf. Fig. 51a), the partitional clustering (step i1 and i2, cf. Fig. 46a) of the training set (cf. Table 39) is first performed for all solutions of the new population (cf. Table 53). The fitness of each solution is then calculated with Eq. (17). The resulting values of D and fitness are reported in Table 54. In Table 55, the new population (generation 1, cf. Table 54) is compared to the initial population (generation 0, cf. Table 50) in terms of minimum and average dispersion values.

Table 54: Partitioning dispersion (D) and fitness of the solutions in the new population (generation 1).

Solution #	D	Fitness
1	204	0.18
2	255	0.15
3	108	0.34
4	184	0.20
5	297	0.13

Table 55: Comparison of the dispersion values of the initial population (generation 0) and of the new population (generation 1).

Generation	D_{min}	D_{av}	ΔD_{min} [%]	ΔD_{av} [%]
0 (initial)	164	223	---	---
1 (new)	108	210	-34	-6

The results indicate that both minimum and average dispersion values decrease from the initial to the new population. In the current generation, the population of solutions has thus evolved towards the partitioning of minimum dispersion.

Stop condition. According to [107,108], the iterative optimization phase is generally stopped based on one or more of the following conditions:

- The achievement of a fixed number of generations (G_{max}, cf. Fig. 51b).
- The achievement of a fixed execution time.
- The achievement of a fixed number of generations without an improvement of the parameter being optimized (as the minimum partitioning dispersion in the population).
- The convergence of the parameter being optimized towards a fixed value.

The first option is adopted in this work. Notably, fixing a sufficiently high G_{max} enables to better monitor and study the optimization phase without potential interruptions. When $G = G_{max}$, the GA is stopped (cf. Fig. 51b). The partitioning of minimum dispersion $P^{K=2}$ and the associated dispersion value D^K (cf. Fig. 46a) are given by the values of $AS1_1$, $AS1_2$, $AS2_1$ and $AS2_2$ corresponding to the solution with the minimum dispersion achieved in the optimization phase.

2.2.3.5. Solution diversity and hyperparameters of the GA

As seen in Fig. 53a, the solutions characterised by a low fitness value are not excluded from parental selection, but they rather have less probability to be selected as parents of the new population. The following questions may thus be spontaneous. Why should bad solutions be included in parental selection? Is it not better to consider only good solutions? As argued in [107,108], the genetic optimization efficiency strongly depends on the diversity of the solutions in the population. At each generation, the population should be composed of solutions as diverse as possible, thus including good as well as bad ones. This is evidently assured by the presence of also bad solutions among the selected parents. Without selecting any bad solutions, solution diversity in the population decreases, and the GA risks to converge towards apparent optimums. Similarly to ANNs, the GA involves some hyperparameters which are typically set-up to favour and maintain a certain solution diversity in the population over the generations:

- The number of genes per variable (g), which is fixed in the first step of the preliminary phase (solution encoding, cf. Fig. 51a and Fig. 52). As seen in the above example, g determines the number and the entity of the possible solution values (cf. Eq. (16)): the higher g, the higher the number of potentially explorable solutions, and the more varied the population at each generation. However, increasing g also means increasing the duration of the optimization procedure, because the solution space is more refined (the research space is wider), and more generations are generally required to achieve the optimal solution.

- The number of solutions in the population (N), which is fixed in the second step of the preliminary phase (initial population, cf. Fig. 51a). The lower N, the less diverse the couples of parents obtained during parental selection. For example, the couples 2 and 4 in Table 51 involve the same parents (i.e. the solutions 1 and 2). In general, same parents risk to generate identical sons, thus mining the diversity in the population. Otherwise, the higher N, the more the selected couples of parents are characterised by diverse solutions, but the more the time needed to execute the operations of parental selection, reproduction and fitness evaluation at each generation (cf. Fig. 51b). In accordance with [107,108], we also believe that fixing a very high N does not make so much sense. Indeed, the base idea of the GA is to iteratively evolve the population (cf. Table 45) by exploring the solution space little by little (cf. Table 45), without necessarily test a huge number of combinations at each generation.
- Parameters associated to the modified roulette method. Assuring that also bad solutions could become parents of the new population is crucial to maintain a certain solution diversity over the generations. The higher N, the less the roulette method (cf. Fig. 53a) is capable to assure the selection of very bad solutions during parental selection. For example, the roulette method shown in Fig. 55a regards a population composed of 50 solutions. The sections indicated by the red arrows correspond to the worst solution (they are almost "invisible", cf. Fig. 55a). Their fitness values are so low that these solutions have almost no chance of being selected as parents. This has a potential negative effect on solution diversity over the generations, causing the GA to converge towards an apparent optimal solution. The selection of also bad solutions can be assured by adopting a modified roulette method, as the one shown in Fig. 55b. The solutions in the population are first organized in order of increasing dispersion (from the best solution to the worst one). Therefore, the minimum dispersion value (D_{min}) will correspond to solution 1, while the maximum dispersion value (D_{max}) will correspond to solution N. The ordered population is then divided into four partitions, each one composed of N/4 solutions, as illustrated in Table 56. A different fitness is finally assigned to each partition by the user (cf. Fig. 55b). During parental selection, the parents will be randomly selected in the partition indicated by the parent selector [107,108].

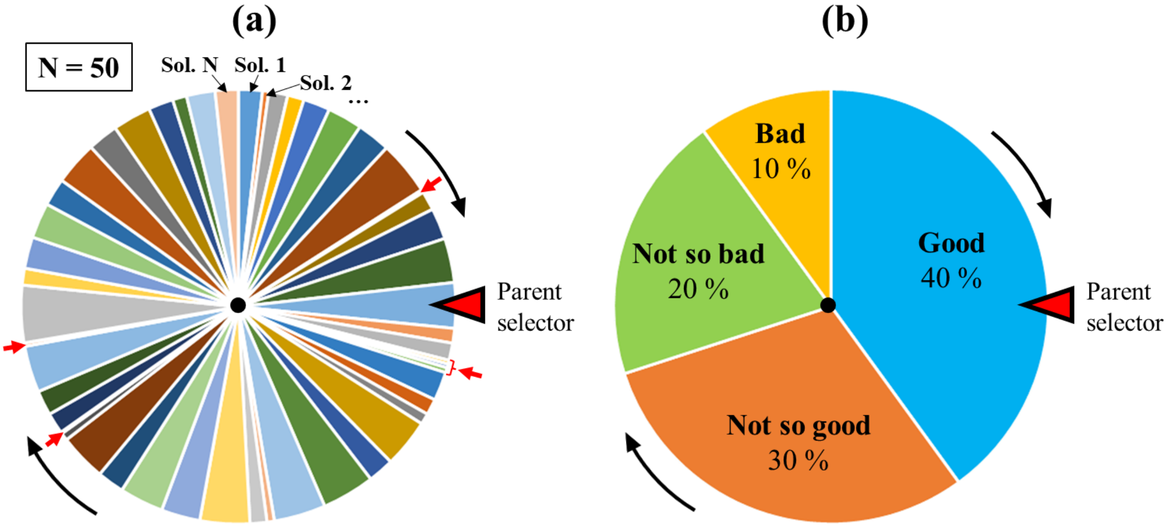


Fig. 55: (a) Example of roulette method (cf. Fig. 53a) with N=50 and (b) modified roulette method.

Table 56: Partition of the ordered population in the modified roulette method (cf. Fig. 55b).

Partition	Solution #	Dispersion value
Good	1	D_{min}
	2	…
	…	…
	N/4	…
Not so good	N/4 + 1	…
	…	…
	2N/4	…
Not so bad	2N/4 +1	…
	…	…
	3N/4	…
Bad	3N/4 + 1	…
	…	…
	N	D_{max}

Similarly to the standard roulette method (cf. Fig. 53a), good solutions have a higher probability to be selected as parents, as well as bad solutions have a lower probability. However, the probability to select also bad solution is no more affected by high values of N (as in Fig. 55a), and this enables to better maintain solution diversity over the generations. In general, the hyperparameters associated to the modified roulette method regard the number of partitions, the number of solutions in each partition and the fitness assigned to each one (cf. Fig. 55b and Table 56).

- The probabilities of cross-over (p_c) and mutation (p_m), which are used to alternatively enable and disable the homonymous mechanisms during reproduction (cf. Fig. 51b). Both p_c and p_m are defined in the range [0;1] and must be fixed before starting the optimization phase. The parameter p_c allows a certain percentage of parents to survive, by directly becoming members of the new population without crossing. For example, if $p_c = 0.7$ (70%), the 30% of solutions in the new population corresponds to surviving parents, randomly chosen among all the couples of parents obtained by the parental selection. Beside the single and the double cross-over (cf. Fig. 54a), p_c thus introduces a new mechanism which further contributes to diversify the solutions in the new population. Similarly, p_m allows only a certain percentage of the generated sons to mutate. For example, if $p_m = 1$ (100%), all the new generated sons mutate. This evidently enables to maintain a high solution diversity in the population, because the mutating genes are always selected at random. However, it risks destroying all the good genetic characteristics transmitted by the parents to the sons, preventing the population to progress towards an optimal solution. For this reason, the mutation occurrence is typically limited by setting p_m lower than 100% [107,108].

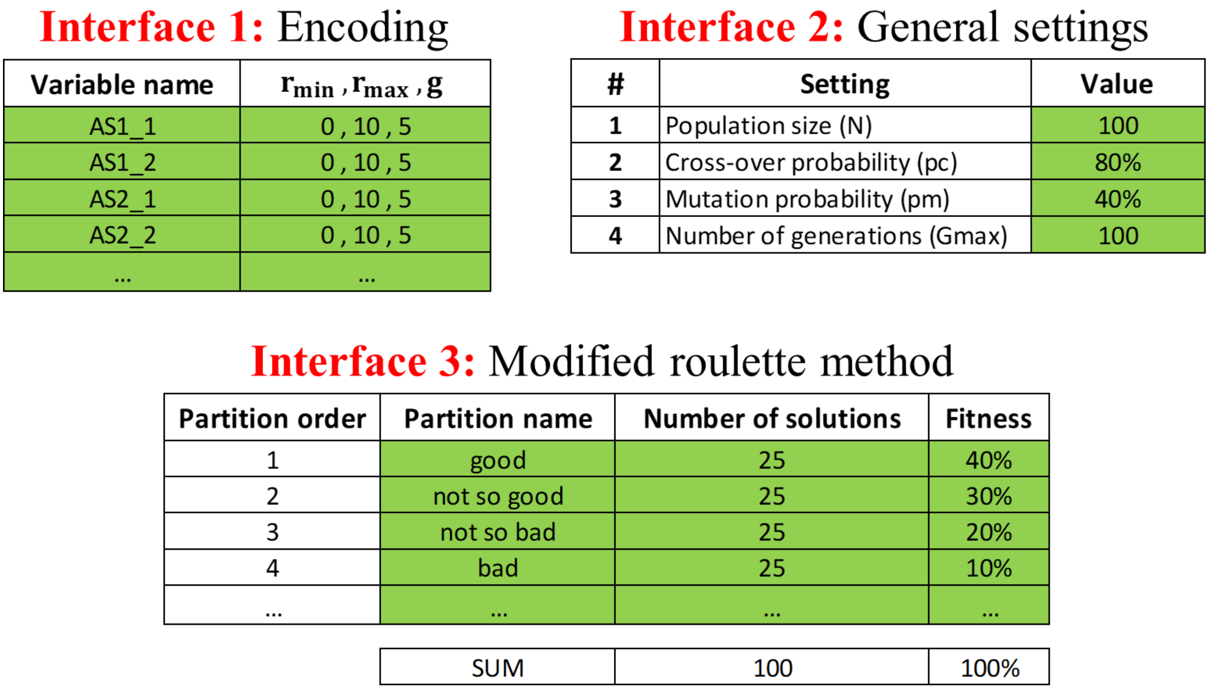
The hyperparameters of the GA and their effects and roles are summarised in Table 57. Similarly to ANN, GA hyperparameters will be fixed according to literature indications and to priorly conducted optimization tests, which for brevity will not be presented in this work.

Table 57: GA hyperparameters and corresponding effects/roles.

Parameter	Effects/Roles
Number of genes per variable (g)	The higher g, the higher the number of potentially explorable solutions. Positive effect: the population is more varied (more diversity). Negative effect: more generations required to achieve the optimal solutions (more time).
Number of solutions in the population (N)	The higher N, the more diverse the selected couples of parents. Positive effect: more diverse solutions generated at each reproduction. Negative effects: • More time to execute the operations of parental selection, reproduction and fitness evaluation (cf. Fig. 51b) at each generation. • The roulette method is less capable to assure the selection of very bad solutions during parental selection (cf. Fig. 55a).
Modified roulette method: • Number of partitions • Number of solutions in each partition • Fitness assigned to each partition	Role: Assuring the selection of also very bad parents. This enable to better maintain a certain solution diversity over the generations.
Cross-over probability (p_c)	Defined between 0 and 1. It allows a certain percentage of parents to directly become members of the new population without crossing. This mechanism contributes to further diversify the solutions in the new population.
Mutation probability (p_m)	Defined between 0 and 1, but typically < 1. It allows only a certain percentage of the generated sons to mutate, avoiding losing all the good characteristics transmitted by parents to the new population at each generation.

2.2.3.6. Presentation and testing of the implemented GA

Based on the theory presented in sections 2.2.3.4 and 2.2.3.5, a versatile model of GA was implemented in MATLAB [48], and it enables to perform the step 1e of the clustering procedure by means of the K-means method (cf. Fig. 46a, b). Similarly to ANNs, the set-up interface of our GA is entirely handled by means of the three Excel tables shown in Fig. 56, which make the GA initialization very intuitive and original. The data contained in the green cells can be modified by the user.



Interface 1: Encoding

Variable name	r_{min} , r_{max} , g
AS1_1	0 , 10 , 5
AS1_2	0 , 10 , 5
AS2_1	0 , 10 , 5
AS2_2	0 , 10 , 5
...	...

Interface 2: General settings

#	Setting	Value
1	Population size (N)	100
2	Cross-over probability (pc)	80%
3	Mutation probability (pm)	40%
4	Number of generations (Gmax)	100

Interface 3: Modified roulette method

Partition order	Partition name	Number of solutions	Fitness
1	good	25	40%
2	not so good	25	30%
3	not so bad	25	20%
4	bad	25	10%
...	...	...	...
	SUM	100	100%

Fig. 56: Interface tables of the implemented GA.

Let us explain the function of each interface table:

- The features of the ASs to be determined (i.e. the variables of the GA) are initialized in the first interface table (interface 1, cf. Fig. 56), in terms of range boundaries (r_{min} and r_{max}, cf. Eq. (16)) and number of genes (g). Clearly, the number of initialized variables depends on the adopted number of clusters (K, cf. Fig. 46a). For example, the four variables “AS1_1”, “AS1_2”, “AS2_1” and “AS2_2” initialized in Fig. 56 regard the partitioning of a training set in R^2 into two clusters (K=2). The ASs to be determined are defined in the range [0;10] and have $2^5 = 32$ possible values spaced of $\Delta r = 0.32$ (cf. Eq. (16)). If K=3 is adopted, for instance, the variables “AS3_1” and “AS3_2” will be simply added in the afterwards rows of interface 1. It is important to remark that interface 1 also enables to choose different values of r_{min}, r_{max} and g for each variable.
- The second interface table (interface 2, cf. Fig. 56) involves the set-up of the hyperparameters N, p_c and p_m (cf. Table 57), and of the maximum number of generations to perform (G_{max}, cf. Fig. 51b). As suggested in [107,108], all these parameters will be fixed according to previous tests to assure the genetic optimization efficiency with a reasonable execution time.
- The third interface table (interface 3, cf. Fig. 56) involves the set-up of the hyperparameters regarding the modified roulette method (cf. Table 57). For example, the values chosen in Fig. 56 correspond to the modified roulette method shown in Fig. 55b and Table 56. A higher number of partitions can be simply set-up by adding, for instance, a fifth partition, and maintaining the total number of solutions equal to N and the sum of the fitness probabilities equal to 100%.

The implemented GA is thus tested in a partitional clustering problem. The considered training set, shown in Fig. 57, consists of 500 training samples in R^2 and it is characterised by three main portions whose centroids correspond to A, B and C. Notably, the point H is the centroid of the overall training set, while $D^{K=1} = 4795$ represents the dispersion of the training samples around the centroid H and it is calculated with Eq. (14). Considering both x and y as CFs (cf. Fig. 47a), the idea is to obtain the curve K-R_D (cf. Fig. 49) by determining the coordinates of the ASs with the implemented GA, and to finally select the most suitable partitioning. Three different optimization tests will be conducted for each K for verifying the reproducibility of the obtainable results (and thus the robustness of the implemented GA[17]), as done for the implemented ANN (cf. section 2.2.2.6). The list of the conducted tests is reported in Table 58.

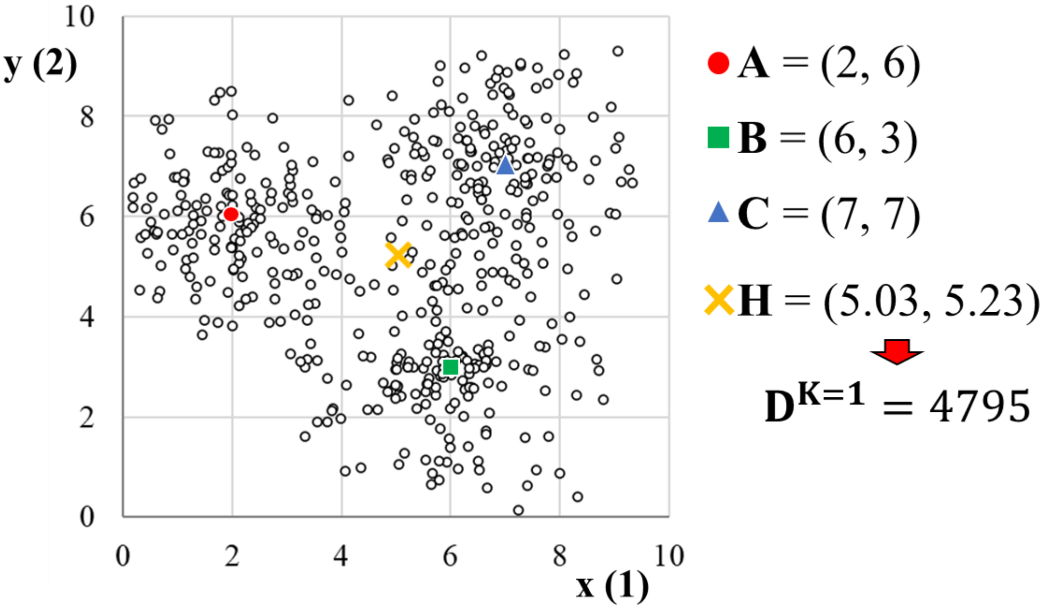


Fig. 57: Considered training set.

Table 58: List of the conducted optimization tests.

Optimization test #	1	2	3	4	5	6	7	8	9	10	11	12
K	2			3			4			5		

Each optimization test involves two distinct steps:

- Set-up, i.e. the initialization of the variables to determine (i.e. “AS1_1”, “AS1_2”, etc.) and of the GA hyperparameters in the program interfaces (cf. Fig. 56), before starting the optimization.
- Optimization, following the procedure reported in Fig. 51.

Each step is described in detail as follows.

Set-up. Regarding interface 1, the same values of r_{min} (=0), r_{max} (=10) and g (=5) are fixed for all variables (cf. Fig. 56) in all optimization tests (cf. Table 58). This assures the same Δr (= 0.32, cf. Eq. (16)) and the same number of possible values (= 2^5 = 32) for all variables and makes the results of all tests comparable to each other. Regarding interface 2, the hyperparameters N, p_c and p_m are fixed as in Fig. 56, while increasing values of G_{max} are adopted depending on K. This choice can be explained with the help of Table 59: the higher K, the higher the number of variables to initialize in interface 1, and the higher the number of genes per chromosome. Consequently, the higher K, the wider the solution space (i.e. the higher the number of possible variable combinations which can be explored during the

[17] The solutions in the initial population are randomly generated (cf. Fig. 51a). Therefore, all tests start with different initial populations. The robustness condition is the following: for each K, the three optimization tests must have similar values of D^K (cf. Fig. 46a), which represents the minimum dispersion achieved in the optimization phase.

optimization, cf. Table 59), and the higher the G_{max} granted to the GA to achieve the optimal solution (cf. Table 57, effect of the number of genes).

Table 59: Adopted values of G_{max} as function of K.

K	Number of variables	Number of genes per variable (g)	Number of genes per chromosome	Number of possible combinations	Adopted G_{max}
2	4	5	20	$2^{20} \approx 10^{6}$	100
3	6	5	30	$2^{30} \approx 10^{9}$	200
4	8	5	40	$2^{40} \approx 10^{12}$	300
5	10	5	50	$2^{50} \approx 10^{15}$	400

Regarding interface 3, the settings of the modified roulette method are fixed as in Fig. 56: four partitions with a selection probability decreasing from 40% to 10%. The fixed values of N, p_c and p_m (interface 2, cf. Fig. 56), the values of G_{max} adopted in Table 59 and the used settings of the modified roulette method (interface 3, cf. Fig. 56) are all based on literature indications (cf. [107,108]) and on several tests conducted with the implemented GA, which for brevity are not presented in this work.

Optimization. After performing the preliminary phase (cf. Fig. 51a), the iterative optimization procedure is launched (cf. Fig. 51a). The GA is stopped when the G_{max} corresponding to the current K is achieved (cf. Table 59).

The results of the conducted optimization tests are firstly presented and compared in terms of D_{min}, which represents the minimum dispersion in the population at each generation (cf. Table 55). The trends of D_{min} observed during the optimization tests are reported in Fig. 58a, b and c. All trends are characterised by several generation ranges where D_{min} remains constant (plateaux), alternating with sudden diminutions of D_{min}. In all tests, the greatest decreases of D_{min} are localised in the first 50 generations. Further less meaningful diminutions can be remarked for the tests with K=3, K=4 and K=5, until the last generation (G_{max}). The minimum dispersion achieved by the GA ($G = G_{max}$, cf. Fig. 58a) corresponds to D^K, i.e. the dispersion of the final partitioning P^K (cf. Fig. 46a). The values of D^K achieved in the optimization tests are summarised in Table 60, together with the coefficients of variation (COV), average values (D^K_{av}) and percentage dispersion reductions (R_D, cf. Eq. (15)) calculated for each K. ΔR_D represents the increment of R_D between two adjacent values of K.

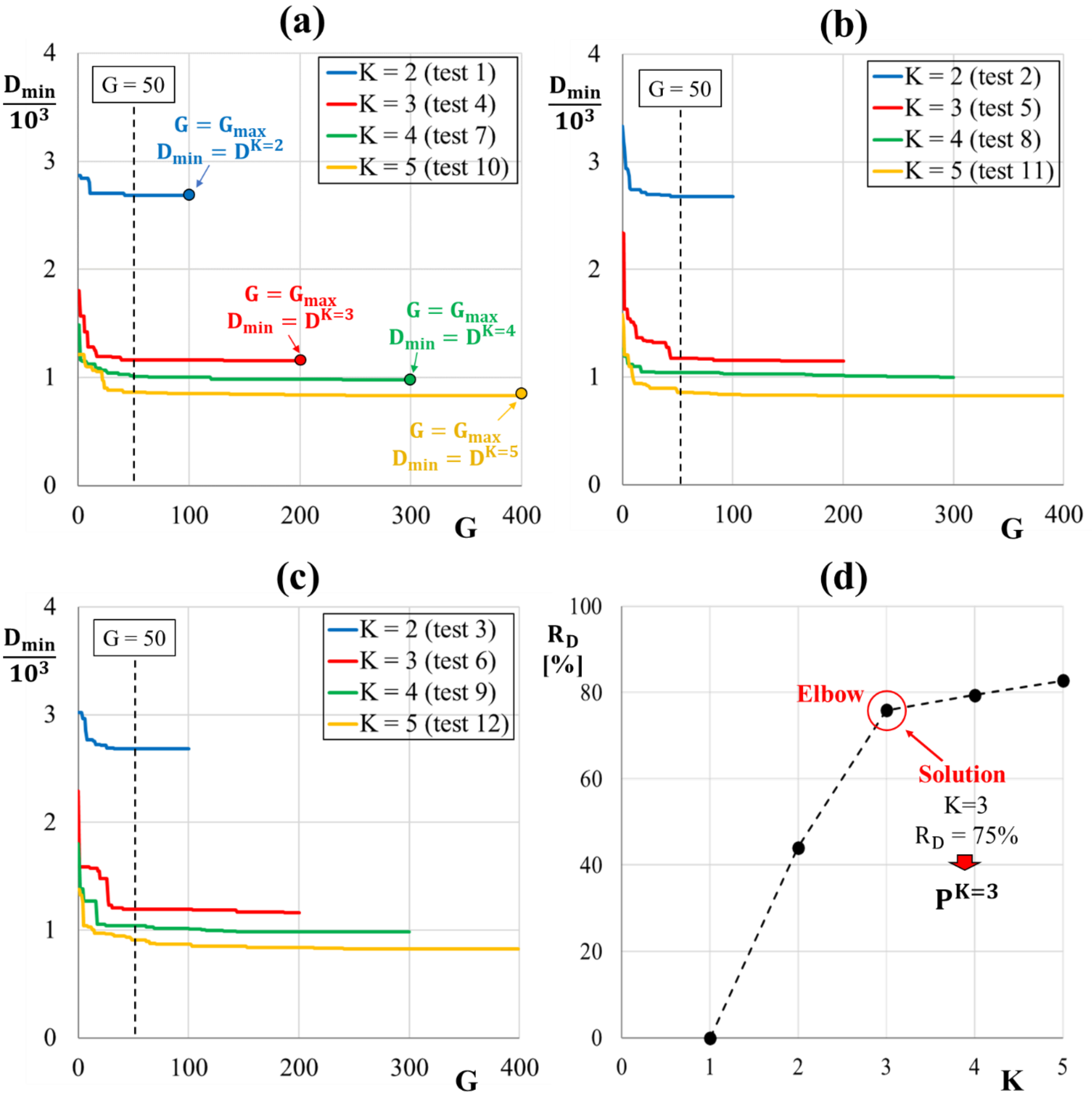


Fig. 58: Trends of D_{min}: (a) tests 1, 4, 7, 10; (b) tests 2, 5, 8, 11; (c) tests 3, 6, 9, 12 (cf. Table 58). (d) Curve K-R_D.

Table 60: Values of D^K achieved in the optimization tests, and variation coefficients (COV), average values ($D^K{}_{av}$) and percentage dispersion reduction (R_D, cf. Eq. (15)) for each K.

K	Test #	D^K	COV [%[	$D^K{}_{av}$	R_D [%]	ΔR_D [%]
1	---	4795	---	4795	0	---
2	1	2682	0.02	2681	44	+44
	2	2681				
	3	2681				
3	4	1153	0.29	1155	75	+31
	5	1153				
	6	1160				
4	7	976	0.97	986	79	+4
	8	999				
	9	984				
5	10	829	0.06	828	83	+4
	11	828				
	12	828				

Two main considerations can be made according to Table 60:

- **About the robustness of the implemented GA**. The factor supporting the robustness of the implemented GA can be deducted by the comparison between the adopted values of G_{max} and the number of possible combinations for each K (cf. Table 59), and the very low variation coefficients obtained for each K (between 0.02% and 0.97%, cf. Table 60). In all the conducted tests, the GA could never have explored all the possible combinations with such a low G_{max}, since the number of explorable combinations is given by the adopted number of solutions in the population (N = 100, cf. Fig. 56) multiplied by G_{max}. For instance, in the tests with K=5, the GA could have explored a maximum of 40,000 different combinations, when the effective number of possible combinations is about 10^{15} (cf. Table 59), and thus extremely higher. However, for each K, all tests converged towards almost the same values of D^K, as suggested by the low variation coefficients in Table 60, despite different initial populations[17] and the adopted $G_{max,}$ which is very low if compared to the total number of potentially explorable combinations. This confirms the robustness of the implemented GA.
- **About the selection of the most suitable partitioning and the relevancy of the resulting clusters**. For each K, the values of R_D reported in Table 60 have been obtained with Eq. (15), by considering the average values of D^K (i.e. D^K_{av}). The corresponding curve K-R_D is represented in Fig. 58d. The more K increases, the more D^K reduces and the more R_D increases. It can be easily remarked that K=2 and K=3 involve the most significant reductions of D^K (respectively ΔR_D=+44% and ΔR_D=+31%, cf. Table 60). A much less meaningful R_D increments are obtained for K>3 (ΔR_D=+4%, cf. Table 60). Based on [104–106], the elbow of the curve K-R_D (cf. Fig. 58d) is neatly visible and suggests that K=3 (and thus $P^{K=3}$) is the best compromise between the minimum dispersion and the minimum number of clusters of the partitioning. The partitioning $P^{K=3}$ adopted as final solution of the clustering procedure is shown in Fig. 59, and it is the one obtained in tests 4 and 5, which achieved the lowest value of D^K for K=3 (i.e. $D^{K=3}$ = 1153, cf. Table 60). The coordinates of the corresponding ASs determined with the GA are reported in Table 61, and are all multiples of Δr = 0.32, which results from Eq. (16) with the set values of r_{min}, r_{max} and g (cf. Fig. 56, interface 1).

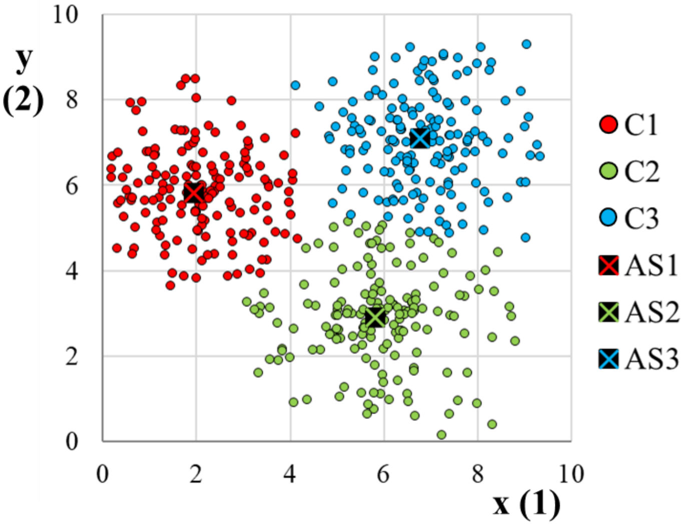


Fig. 59: $P^{K=3}$ (tests 4 and 5).

Table 61: Coordinates of the ASs in Fig. 59.

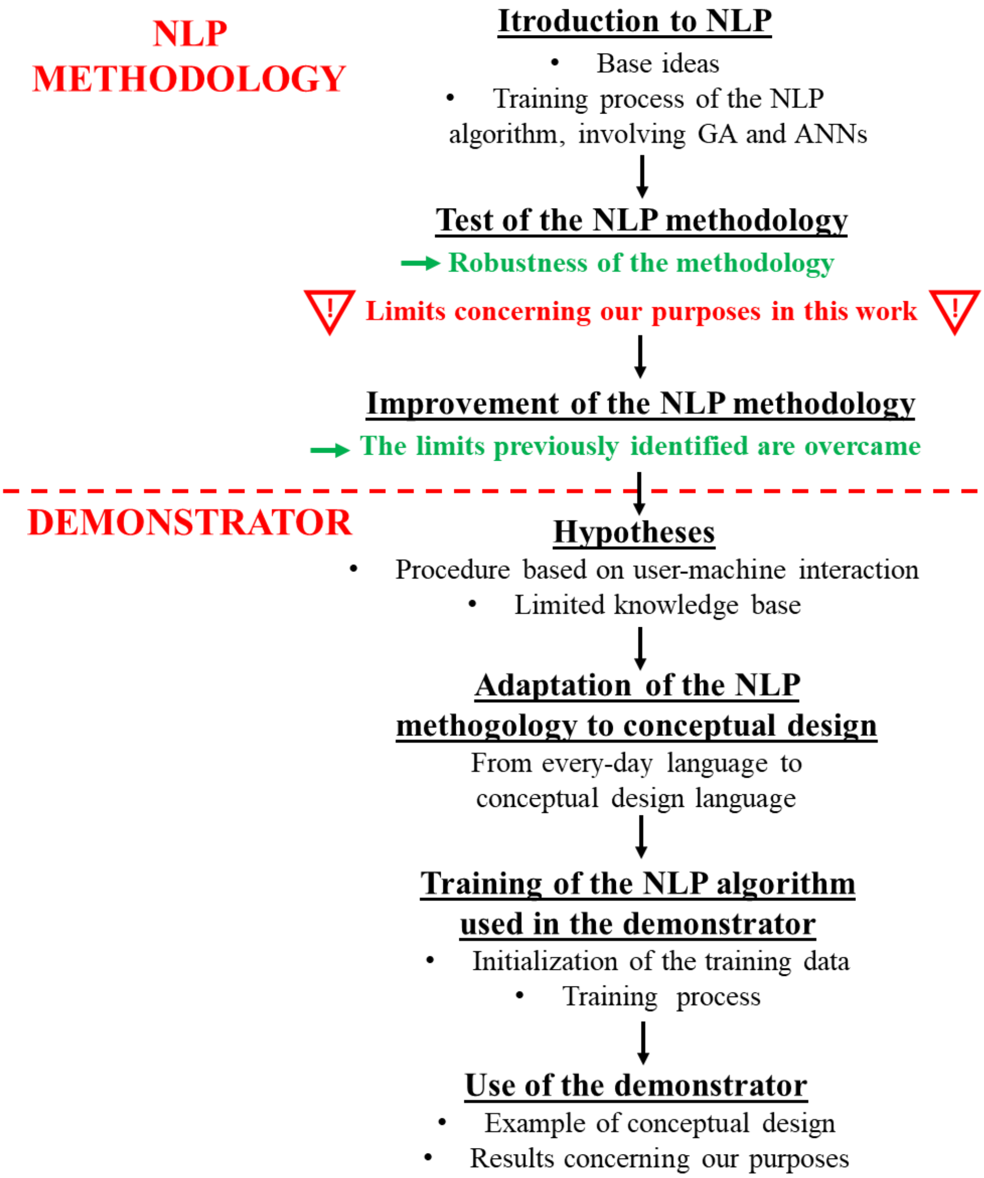
	x (1)	y (2)
AS1	1.92	5.76
AS2	5.76	2.88
AS3	6.72	7.04

Fig. 59 enables to verify the relevancy of the resulting clusters, in accordance with Table 28 (see unsupervised learning). We can easily remark that the clusters C1, C2 and C3 represent the three main portions of the training set (cf. Fig. 57). Moreover, the coordinates of the ASs in Table 61 are quite similar to the ones of the centroids A, B and C in Fig. 57, and this assures the relevancy of the results provided by the implemented GA with the used clustering method and to the adopted CFs[18].

[18] As seen in section 2.2.3.3, the ASs associated to the partitioning of minimum dispersion (P^K, cf. Fig. 46a) tend to coincide with the centroids of the corresponding clusters (cf. Fig. 48 and Table 43).

3. A natural language processing methodology for conceptual design: an original demonstrator

In the previous chapter, we introduced the reasons for adopting machine learning in the current work (cf. section 2.1), and we then presented the developed ML methods, i.e. ANNs and partitional clustering via GA (cf. section 2.2). According to the adopted approach (PAAs, cf. Table 18), in this chapter we will see how these methods were used for implementing an original methodology of natural language processing (NLP), operating with an everyday and non-scientific language. This NLP methodology was then employed as base for developing our original demonstrator of conceptual design procedure. The conceptual map reported in Fig. 60 represents the path followed in this chapter.



NLP METHODOLOGY
Itroduction to NLP
• Base ideas
• Training process of the NLP algorithm, involving GA and ANNs
Test of the NLP methodology
→ Robustness of the methodology
! Limits concerning our purposes in this work !
Improvement of the NLP methodology
→ The limits previously identified are overcame
DEMONSTRATOR
Hypotheses
• Procedure based on user-machine interaction
• Limited knowledge base
Adaptation of the NLP methogology to conceptual design
From every-day language to conceptual design language
Training of the NLP algorithm used in the demonstrator
• Initialization of the training data
• Training process
Use of the demonstrator
• Example of conceptual design
• Results concerning our purposes

Fig. 60: Conceptual map of the chapter developed to explain the original path leading to implemented demonstrator.

As shown in Fig. 60, the current chapter is organized in two main sections. The first one (section 3.1) is devoted to the NLP methodology, whose base ideas are immediately clarified, and the training process of the NLP algorithm is then described. According to Fig. 60, the afterwards testing of the developed NLP methodology enables to verify its robustness and to outline some limits concerning our purposes in this work (cf. Table 9). The NLP methodology is finally improved to overcome the identified limits. As shown in Fig. 60, the second section of the chapter (section 3.2) is devoted to the demonstrator. The hypotheses on which the demonstrator is based are immediately presented. The NLP methodology previously developed operates with an every-day and non-scientific language, and its adaptation to the conceptual design language is thus described. After training the NLP algorithm, the use of the demonstrator is illustrated with an original example of conceptual design, and the main original results concerning our purposes in this work are finally outlined.

3.1. An original methodology for processing every-day language

As reported in Table 18, the use of the natural language to express and identify the SPs and the CVs represents one of the principles of the adopted approach. In this section, we make the last step before presenting our demonstrator, by describing how the implemented ML algorithms (cf. chapter 3) are combined into NLP methodology. The presented NLP methodology is original, and it operates with an everyday and non-scientific language. This will enable to better understand the base ideas and the technical aspects involved in a NLP algorithm, before introducing a more specific language of mechanical design. The current section is organized as follows. In section 3.1.1, we present the base ideas and the development of our NLP methodology, with a particular emphasis on the training process, which involves the use of the implemented ML methods (ANNs and GA). In section 3.1.2, our NLP methodology is tested for verifying its robustness and for better outlining its limits concerning the purposes pursued in this work (cf. Table 9). In section 3.1.3, the developed NLP methodology is finally improved for resolving the limits highlighted in the previous section.

3.1.1. Base ideas and development of the NLP methodology

The natural language processing involves the machine elaboration of text inputs written in a spoken language [109,110]. According to [109–111], the NLP methodology developed in this work combines the implemented ML methods to accomplish two base ideas, which are schematized in Fig. 61a:

- Understanding the context of an incomplete input sentence by analysing its lexicon and its syntax, for identifying the involved circumstance. The definitions of lexicon and syntax are reported in Table 62.
- Proposing multiple continuations of the incomplete input sentence which are coherent to the identified context.

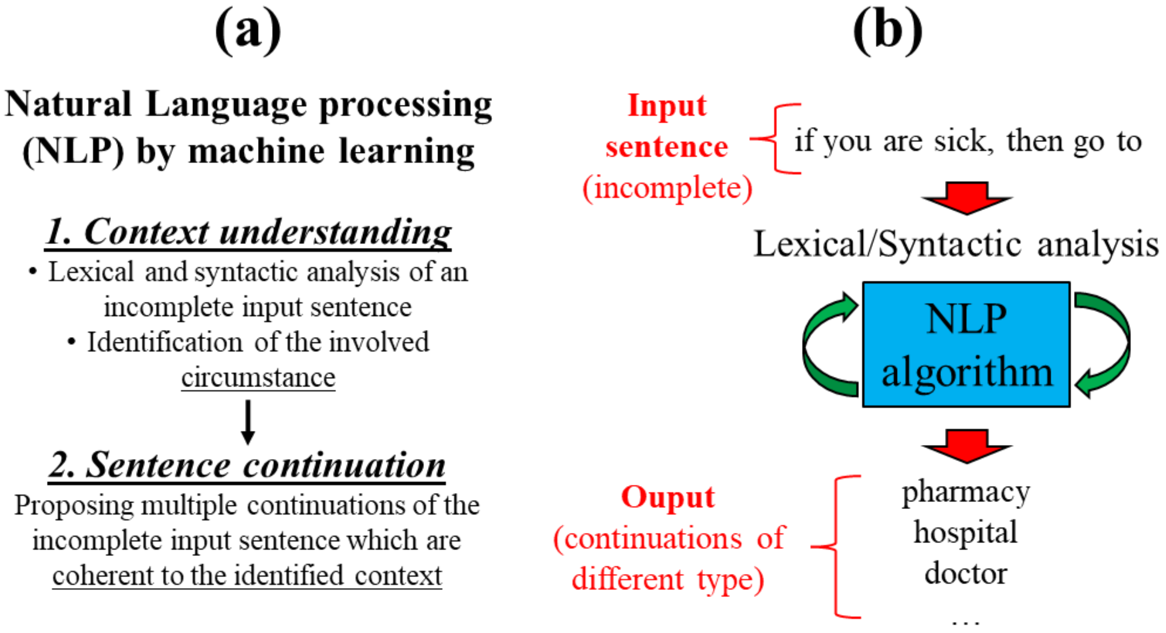

(a)
Natural Language processing (NLP) by machine learning
1. Context understanding
• Lexical and syntactic analysis of an incomplete input sentence
• Identification of the involved circumstance
2. Sentence continuation
Proposing multiple continuations of the incomplete input sentence which are coherent to the identified context
(b)
Input sentence (incomplete)
if you are sick, then go to
Lexical/Syntactic analysis
NLP algorithm
Ouput (continuations of different type)
pharmacy
hospital
doctor
...

Fig. 61: (a) Base ideas of the developed NLP methodology; (b) example of the NLP algorithm functioning.

Table 62: Definitions of lexicon and syntax.

Term	Definition
Lexicon (or vocabulary)	Ensemble of different words used in a sentence (or in a set of sentences)
Syntax	Study of how different words (or lexical elements) are combined (associated and organized) to form sentences of complete sense

In the example reported in Fig. 61b, the incomplete input sentence "if you are sick, then go to" is lexically and syntactically analysed by the NLP algorithm, which then proposes different continuations. All the continuations are coherent with the context of "being sick" and of "going somewhere". Moreover, remark that the proposed continuations are of different type: "pharmacy" and "hospital" identify two specific places, while "doctor" identifies a professional figure, and thus a person. Referring to the adopted conceptual design procedure (step 1, cf. Fig. 16), it is not difficult to imagine the incomplete input sentence as the ensemble of initial idea and specifications (context), and the proposed continuations as multiple and different SPs (see BDPs, cf. Table 5). The abilities of context understanding and sentence continuation (cf. Fig. 61a) are acquired by the NLP algorithm with a suitable training, exploiting the potential of the implemented machine learning methods (GA and ANN). According to [110,111], the training process of the NLP algorithm is realised in two steps, called pre-training and fine-training, and it is schematized in Fig. 62. To avoid creating confusion in the terminology, the term "training" refers to the overall training of the NLP algorithm, while the term "calibration" refers to the training of the ANN in the fine-training step (cf. Fig. 62).

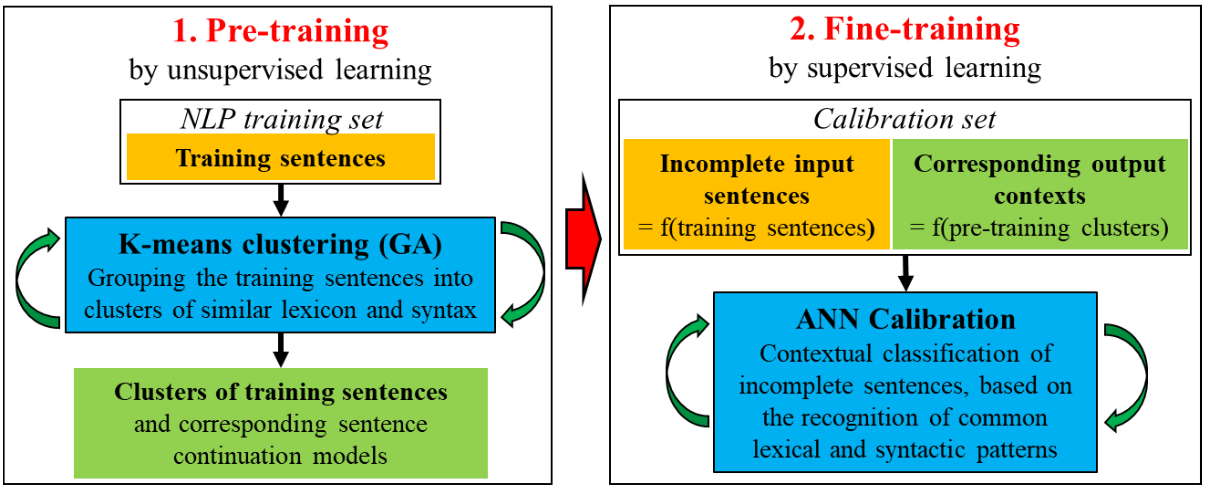

1. Pre-training
by unsupervised learning
NLP training set
Training sentences
K-means clustering (GA)
Grouping the training sentences into clusters of similar lexicon and syntax
Clusters of training sentences
and corresponding sentence continuation models
2. Fine-training
by supervised learning
Calibration set
Incomplete input sentences
= f(training sentences)
Corresponding output contexts
= f(pre-training clusters)
ANN Calibration
Contextual classification of incomplete sentences, based on the recognition of common lexical and syntactic patterns

Fig. 62: Steps of the NLP algorithm training and respective machine learning methods.

As shown in Fig. 62, the pre-training step is based on the unsupervised learning model (cf. Fig. 30a and Table 28), and it involves the K-means clustering of the training sentences, which represent the ensemble of language knowledge we want the NLP algorithm to learn. The idea is to group the training sentences into clusters containing similar lexical and syntactic forms. According to Fig. 62, each cluster will be characterised by a specific sentence continuation model. The following fine-training step is based on the supervised learning model (cf. Fig. 29a and Table 28) and it concerns the calibration of an ANN, which must learn to classify incomplete sentences (as that in Fig. 61b) based on the recognition of common lexical and syntactic patterns. This enables the ANN to acquire the ability of context understanding (cf. Fig. 61b). As reported in Fig. 62, the calibration set consists of incomplete input sentences, generated from the training sentences, and of the corresponding output contexts, which depend on the clusters identified in the pre-training step. After completing both training steps, the NLP algorithm will be finally able to understand the context of new incomplete input sentences via the calibrated ANN (generalization, cf. Fig. 29a), and to propose multiple and different continuations by means of the sentence continuation models constructed in the pre-training step (cf. Fig. 62). The pre-training step is described in sections 3.1.1.1 and 3.1.1.2, while the fine-training step in sections 3.1.1.3 and 3.1.1.4. The scheme of trained NLP algorithm is finally presented in section 3.1.1.5.

3.1.1.1. Pre-training: numerical representation of text inputs

The use of the K-means clustering method (cf. Fig. 46) requires training sentences are transformed from textual into numerical inputs[19]. Based on Fig. 62, the numerical representation of text inputs must conserve both lexical and syntactic characteristics of the training sentences (cf. Table 62). Let us consider, for example, the set of three sentences reported on the left side of Table 63 as our training input (cf. Fig. 62), where each sentence thus corresponds to a different training sample. The bag-of-words model (BOW), shown in Table 63, enables to easily transform textual inputs into numerical ones, and it is one of the most used in NLP [112,113]. In this model, the training set is represented based on different features (cf. Table 9), which correspond to the words of the overall used lexicon (i.e. "you", "like", …, "with", cf. Table 63). Notably, each sentence is expressed as a binary array, where the values 1 and 0 respectively indicate the presence or the absence of the word in the sentence.

Table 63: Set of considered training sentences and corresponding BOW model based on word presence/absence.

#	Sentence	BOW (word presence)								
		you	like	football	and	I	love	music	eat	with
S1	you like football	1	1	1	0	0	0	0	0	0
S2	you and I love music	1	0	0	1	1	1	1	0	0
S3	I eat with you	1	0	0	0	1	0	0	1	1

This representation effectively describes the lexicon used in each sentence (i.e. which words are used, cf. Table 62), but it completely ignores the syntax (i.e. how words are combined, cf. Table 62), which is however needed for a more accurate definition of the context (cf. Fig. 61a). An enhanced BOW model able describe the sentence syntax can be obtained by considering the relative position of the words in each sentence, instead of their presence. The proposed enhanced BOW model represents an original improvement of the standard BOW described in [112,113] (cf. Table 63). Let us indicate with N_w the number of words in the generic sentence. Therefore, let us consider the generic sentence as a segment of length 1, divided into N_w portions of length Δx. The expression of Δx is reported in Eq. (18). The relative position of the i^{th} word in the generic sentence (x_i) can be finally calculated with Eq. (19).

$$\Delta x = \frac{1}{N_w}. \tag{18}$$

$$x_i = \begin{cases} \Delta x/2 & \text{for } i = 1 \\ x_{i-1} + \Delta x & \text{for } i > 1 \end{cases}. \tag{19}$$

For example, Fig. 63 illustrates how Eqs. (18) and (19) are applied to the sentence S1 (cf. Table 63). The enhanced BOW model based on word relative positions is reported in Table 64, which also indicates the frequency of each word of the overall lexicon in the training sentences. The most frequent words, i.e. "you" and "I", have been highlighted in red.

[19] The use of the similarity and dispersion measures (cf. Eqs. (13) and (14)) in the clustering procedure requires numerical inputs.

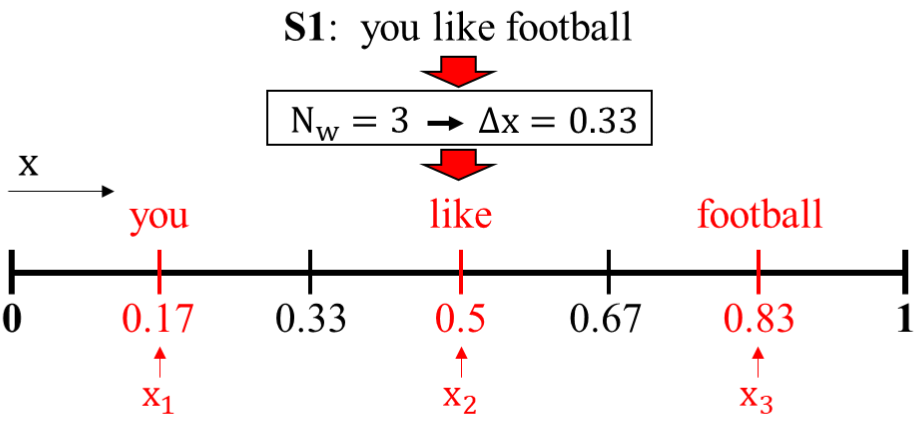


Fig. 63: Calculation of the word relative positions for S1 (cf. Table 63 and Eqs. (18), (19)).

Table 64: Set of considered training sentences and corresponding enhanced BOW model based on word position (cf. Table 63).

#	Sentence	Enhanced BOW (word relative position)								
		you	like	football	and	I	love	music	eat	with
S1	you like football	0.17	0.5	0.83	0	0	0	0	0	0
S2	you and I love music	0.1	0	0	0.3	0.5	0.7	0.9	0	0
S3	I eat with you	0.88	0	0	0	0.13	0	0	0.38	0.63

Presence Frequency	3	1	1	1	2	1	1	1	1

Contrary to the BOW model based on word presence (cf. Table 63), the enhanced BOW model describes both the employed lexicon (non-zero values, cf. Table 64) and the syntax (relative order of the used words) in the training sentences.

3.1.1.2. Pre-training: K-means clustering and construction of the sentence continuation models

The K-means method (cf. Fig. 62) can now be applied by considering each training sentence as a point in a n-dimensional space, whose coordinates correspond to the relative positions of the words (cf. Table 64). Based on [111,114], the most frequent words ("you" and "I", cf. Table 64) can be considered as the most representative features of the training set, and are thus adopted as CFs. Notably, adopting few representative features (rather than all the available ones) as CFs involves few ASs to determine, and it thus enables to conduct the clustering procedure in a reasonable time. The training samples to cluster are resumed in Table 65. In this case, given the small size of the training set, the clustering procedure (cf. Fig. 46) was directly performed with Eqs. (13) and (14), without using the GA. The results in terms of D^K and R_D are reported in Table 66 and Fig. 64a. For K=3, the number of clusters equals the number of training samples, and this clearly gives $R_D = 100\%$ (cf. Table 66). As indicated by the elbow in Fig. 64a, the partitioning $P^{K=2} = \{C1,C2\}$ is the solution of the clustering procedure[20] and it is represented in Fig. 64b, while the corresponding ASs are provided in Table 67.

[20] Remember that K must be minimized (cf. Fig. 46a).

Table 65: Samples to cluster (Table 64)

S#	you	I
S1	0.17	0
S2	0.1	0.5
S3	0.88	0.13

Table 66: Values of D^K and R_D (cf. Eq. (15))

K	D^K	R_D [%]
1	0.51	0
2	0.13	75
3	0	100

Table 67: ASs' coordinates in $P^{K=2}$.

	you	I
AS1	0.14	0.25
AS2	0.88	0.13

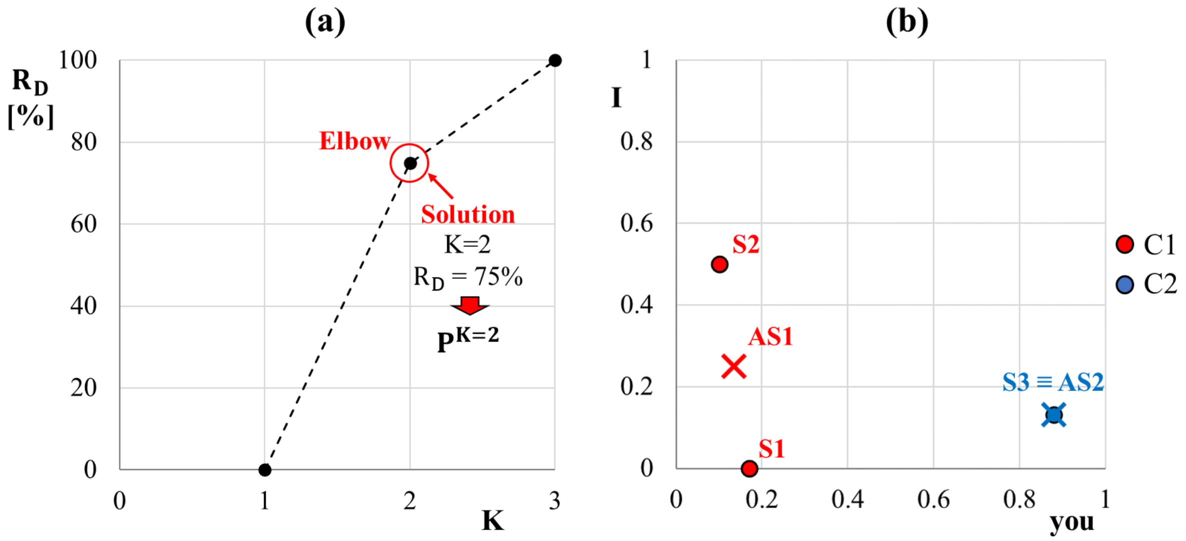
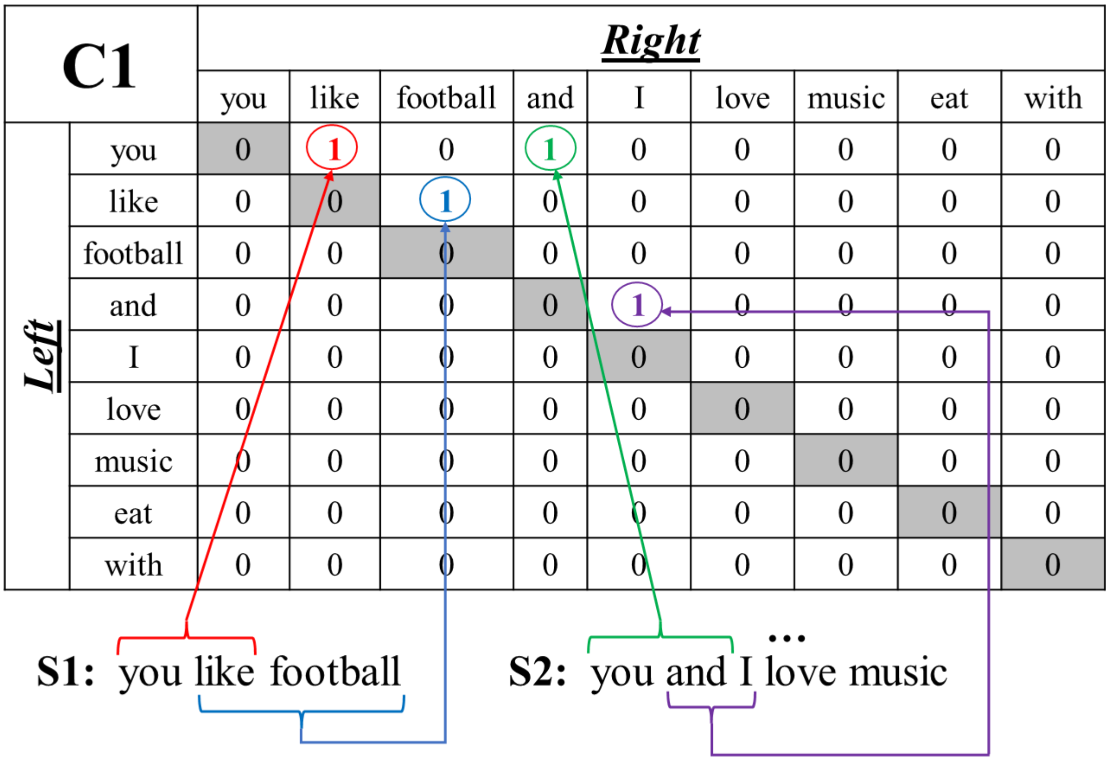


Fig. 64: (a) Curve K-R_D and (b) representation of the partitioning $P^{K=2}$ = {C1, C2} (CFs = "you", "I").

We can easily remark that AS1 is the centroid of the cluster C1 = {S1,S2}, while AS2 coincides with S3 since the latter is the unique sample contained in the cluster C2 (cf. Fig. 64b). According to Fig. 62 (pre-training), we grouped the training sentences into clusters of similar lexicon and syntax, where the similarity is clearly based on the adopted CFs ("you" and "I", cf. Table 65). Now, we must construct the sentence continuation model of each cluster. The construction procedure of the sentence continuation model for the cluster C1 (cf. Fig. 64b) is shown in Fig. 65. The continuation model of C1 is firstly initialized as a square matrix of zeros, whose rows and columns contain all the features of the considered training set (i.e. the 9 different words of the overall lexicon, cf. Table 64), while the grey cells indicate the diagonal of the matrix. Sentence by sentence, the value 1 is then assigned to each couple of words, as represented in Fig. 65. For repeated specification-SP couples, the value 1 in the target cell is maintained. This continuation model is called left-to-right (LTR), since the couples of words are selected by scanning the sentences from left to right [115]. The LTR model of the cluster C2 is constructed in a similar way, as function of the sentence S3. The complete LTR models of C1 and C2 are respectively reported in Table 68 and Table 69, where values 1 have been highlighted in red.



C1		*Right*								
		you	like	football	and	I	love	music	eat	with
Left	you	0	1	0	1	0	0	0	0	0
	like	0	0	1	0	0	0	0	0	0
	football	0	0	0	0	0	0	0	0	0
	and	0	0	0	0	1	0	0	0	0
	I	0	0	0	0	0	0	0	0	0
	love	0	0	0	0	0	0	0	0	0
	music	0	0	0	0	0	0	0	0	0
	eat	0	0	0	0	0	0	0	0	0
	with	0	0	0	0	0	0	0	0	0

S1: you like football

S2: you and I love music

Fig. 65: Construction of the LTR model of sentence continuation for the cluster C1 = {S1, S2}.

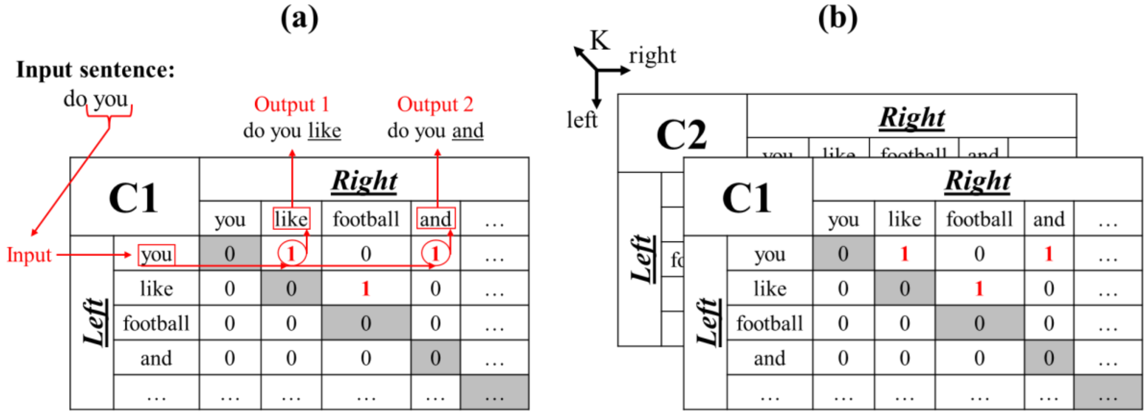
Table 68: LTR model of the cluster C1 = {S1,S2} (cf. Table 64).

C1		***Right*: possible continuation**								
		you	like	football	and	I	love	music	eat	with
***Left*: last word of the sentence to be continued**	you	0	1	0	1	0	0	0	0	0
	like	0	0	1	0	0	0	0	0	0
	football	0	0	0	0	0	0	0	0	0
	and	0	0	0	0	1	0	0	0	0
	I	0	0	0	0	0	1	0	0	0
	love	0	0	0	0	0	0	1	0	0
	music	0	0	0	0	0	0	0	0	0
	eat	0	0	0	0	0	0	0	0	0
	with	0	0	0	0	0	0	0	0	0

Table 69: LTR model of the cluster C2 = {S3} (cf. Table 64).

C2		***Right*: possible continuation**								
		you	like	football	and	I	love	music	eat	with
***Left*: last word of the sentence to be continued**	you	0	0	0	0	0	0	0	0	0
	like	0	0	0	0	0	0	0	0	0
	football	0	0	0	0	0	0	0	0	0
	and	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	0	0	0	1	0
	love	0	0	0	0	0	0	0	0	0
	music	0	0	0	0	0	0	0	0	0
	eat	0	0	0	0	0	0	0	0	1
	with	1	0	0	0	0	0	0	0	0

Based on [115], each LTR model condensates all the information concerning the lexicon and the syntax of the corresponding cluster in a unique representation, without any distinction among the original sentences. This enables to account for different and multiple continuation options when the last word of the incomplete input sentence (cf. Fig. 61b) is part of the training set lexicon. For example, let us consider the incomplete sentence "do you" as input, and let us use the LTR model of C1 (Table 68) to find some possible continuations. Remark that the word “do” is not included in the lexicon of the training set. As shown in Fig. 66a, the last word of the input sentence (“you”, which belongs to the lexicon of the training set) is fed into the LTR model, providing two possible continuations (“like” and “and”) which lead to the output sentences “do you like” and “do you and”. Both output sentences can be further continued by exploiting the same LTR model (cf. Table 68), finally obtaining the sentences “do you like football” and “do you and I love music”. Although these sentences are very similar to S1 and S2, respectively, they are new because they are not included in the original training set (cf. Table 64).



C1	Right				
Left	you	like	football	and	…
you	0	1	0	1	…
like	0	0	1	0	…
football	0	0	0	0	…
and	0	0	0	0	…
…	…	…	…	…	…

C1	Right				
Left	you	like	football	and	…
you	0	1	0	1	…
like	0	0	1	0	…
football	0	0	0	0	…
and	0	0	0	0	…
…	…	…	…	…	…

Fig. 66: (a) Example of sentence continuation by using the LTR model of C1 (cf. Table 68); (b) global three-dimensional LTR database of the obtained partitioning ($P^{K=2}$, cf. Fig. 64).

The mechanism of sentence continuation described in Fig. 66a enables the NLP algorithm to process incomplete sentences whose lexicon is partially unknown, and to generate new sentences which are not included in the original training set. Before passing to fine-training, the constructed LTR models (i.e. Table 68 and Table 69) are organized in the global three-dimensional LTR database represented in Fig. 66b, where the third dimension corresponds to the number of clusters (K) in the obtained partitioning ($P^{K=2}$, cf. Fig. 64).

3.1.1.3. Fine-training: generation of the calibration set and ANN calibration

The calibration set of the ANN consists of different calibration samples, each one composed by an incomplete input sentence and by the corresponding output context. According to Fig. 62, the calibration samples are generated as function of the training sentences (cf. Table 64) and of the clusters identified in the previous pre-training (cf. Fig. 64b). The generation process of the generic calibration sample is illustrated in Fig. 67a, and it is described as follows. An initial training sentence is randomly selected from the training set (phase 1), as the sentence S1 in Fig. 67a (cf. Table 64). The selected sentence is thus cut in a random point between two words and its right end is deleted (phase 2), obtaining an incomplete sentence. In the final step, each word of the current incomplete sentence has the 15% probability to be masked (phase 3). This technique is inspired to the new NLP algorithm recently developed by Google (cf. [111]), and its fundamental functions in our NLP methodology will be discussed in detail in section 3.1.1.4. As shown in Fig. 67a, the resulting calibration sample is thus characterised by the just generated incomplete sentence (input), with potentially masked words, and by its output context, which corresponds to the cluster containing the initial training sentence (i.e. C1, cf. Table 64 and Fig. 64b).

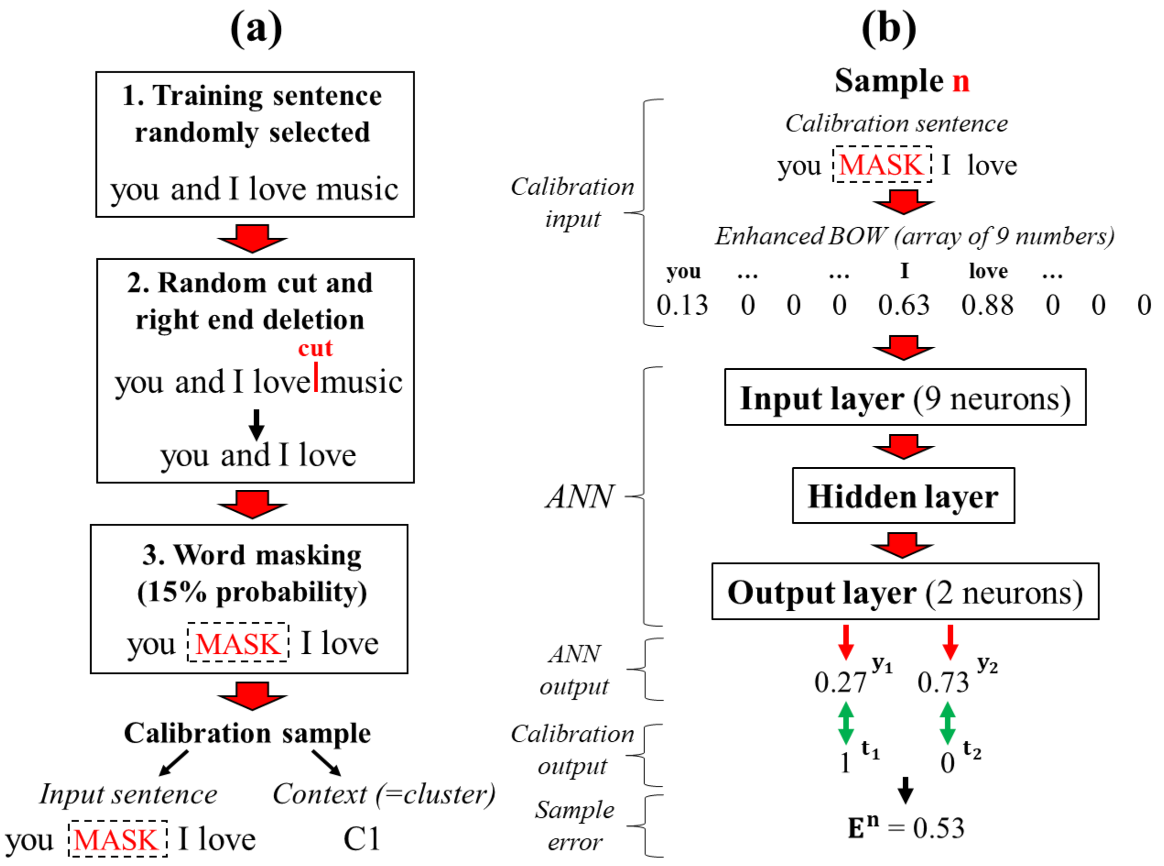


Fig. 67: (a) Generation process of the calibration samples; (b) forward step of the sample n (cf. Table 70).

An example of calibration set generated with the process in Fig. 67a is reported in Table 70, where both mask-with and mask-free sentences are included thanks to the chosen word masking probability (cf. Fig. 67a). For each calibration sentence, the corresponding enhanced BOW model is obtained by Eqs. (18) and (19), and the relative position of the words replaced by "MASK" is zero. For example, if we compare samples 2 and 4 (cf. Table 70) which derive from the same training sentence (S1, cf. Table 64), we can see that the relative position of "you" is 0.25 in sample 2, while it is zero in sample 4 due to masking. Finally, the output context in Table 70 are represented as probability arrays, where t_1 and t_2 are the probabilities of belonging to C1 and C2, respectively (cf. Table 33).

Table 70: Examples of generated calibration samples with the respective enhanced BOW models and probability arrays.

	Calibration input						**Calibration output**		
Sample #	*Incomplete sentence*	*Original sentence*	*Enhanced BOW (9 words)*				*Context (cluster)*	*Probability*	
			you	**like**	**…**	**with**		**t_1**	**t_2**
1	I MASK with	S3	0	0	…	0.83	C2	0	1
2	you like	S1	0.25	0.75	…	0	C1	1	0
3	I eat	S3	0	0	…	0	C2	0	1
4	MASK like	S1	0	0.75	…	0	C1	1	0
…	…	…	…	…	…	…	…	…	…
n	you MASK I love	S2	0.13	0	…	0	C1	1	0
…	…	…	…	…	…	…	…	…	…
N	you and MASK	S2	0.17	0	…	0	C1	1	0

Once the calibration set has been generated, the calibration of the ANN is conducted by following the iterative procedure in Fig. 37a and the tests presented in the section 2.2.2.6. The forward step of the nth calibration sample (cf. Table 70) is illustrated in Fig. 67b, where the enhanced BOW model of the sentence "you MASK I love" represents the calibration input fed into the ANN (cf. Fig. 67b). Clearly, the numbers of input and output neurons correspond to the lengths of the BOW models and of the probability arrays (i.e. 9 and 2, respectively, cf. Table 70), while the number of hidden neurons must be determined iteratively (cf. section 2.2.2.6). As shown in Fig. 67b, the ANN outputs (y_1 and y_2) and the calibration outputs (t_1 and t_2) are thus used to calculate the sample error E^n (cf. Eq. (9)), which will be used in the backword step to calculate weights' corrections (cf. Fig. 37a). For brevity, we assume our ANN is calibrated.

3.1.1.4. Fine-training: the fundamental functions of word masking

Before proceeding, it is important to better understand the crucial role of masked words in the calibration set. It must be said that masking some words during the generation of the calibration samples is a commonly used technique in the training of the NLP algorithms, as the recent bidirectional encoder representation transformer (BERT) developed by Google in 2019 [111]. Based on BERT, the masking word technique has three main functions:

- Generating a higher number of different calibration samples, in accordance with the fundamental guide principles of ANN calibration reported in Table 35. As seen in Table 70, the masking word probability (cf. Fig. 67a) enables to generate both mask-with and mask-free calibration sentences. For example, starting from the training sentence "you and I love music" (cf. Fig. 67a), we can obtain not only the calibration sentences "you and I love", "you and I" and "you and", but also their respective masked versions "you MASK I love", "you and MASK" and "MASK and". This enables to generate a calibration set that is much larger and much more varied than the initial training set.
- Teaching the ANN to process input sentences whose lexicon is partially unknown, since the masked word simulates a word which is not included in the lexicon of the considered training set (as the word "do" in Fig. 66a). Within a certain limit, the use of masked words thus enables to extend the ANN abilities of context understanding without enlarging the original training set.
- Teaching the ANN to process and classify input sentences which are potentially characterised by missing information. Considering the example in Fig. 27, let us suppose masking a word of the initial idea, obtaining the sentence "cantilever MASK full squared section", where the masked word simulates a missing information. The presence of both mask-with and mask-free sentences in the calibration set (cf. Table 70) thus teaches the ANN to work with both weakly and clearly defined inputs.

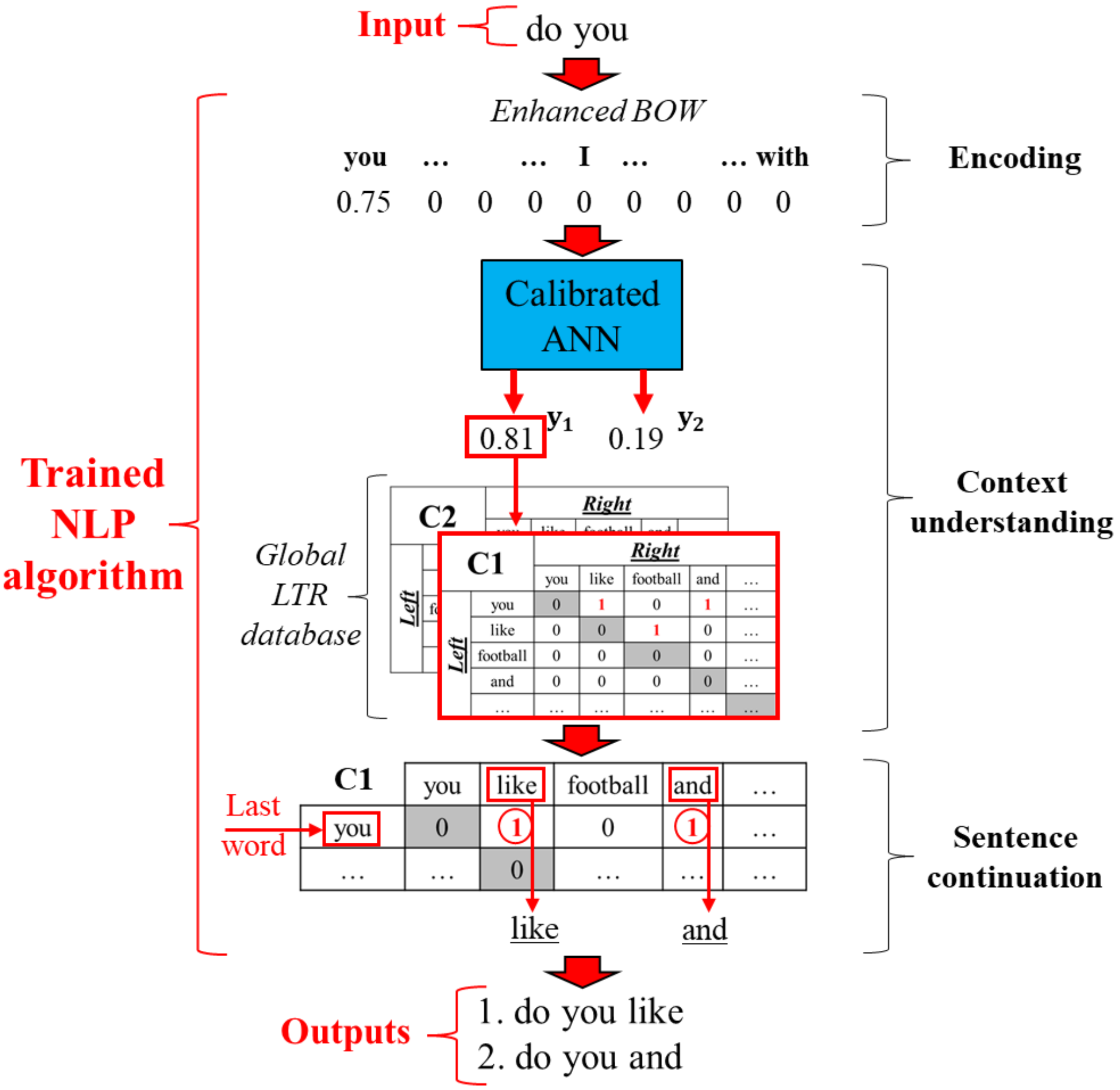
The three functions of the word masking technique are summarised in Table 71. The probability value of 15% (phase 3, cf. Fig. 67a) is adopted according to [111] and to previously conducted tests, which will not be presented for brevity.

Table 71: Functions of the masking word technique (cf. [111]).

#	Function
1	Generating a higher number of different calibration samples. The calibration set is much larger and much more varied than the initial training set.
2	Teaching the ANN to process sentences whose lexicon is partially unknown. This means extending its abilities of context understanding without enlarging the original training set.
3	Teaching the ANN to process also inputs characterised by missing information. The calibrated ANN will be able to successfully elaborate both weakly and clearly defined inputs.

3.1.1.5. Scheme of the trained NLP algorithm

The functioning scheme of the trained NLP algorithm is reported in Fig. 68, and it combines the global LTR database (cf. Fig. 66b) and the calibrated ANN (cf. section 3.1.1.3), respectively obtained in the steps of pre-training and fine-training (cf. Fig. 62). By considering the incomplete input sentence “do you” (cf. Fig. 66b), let us see how the NLP algorithm accomplish the two base ideas initially schematized in Fig. 61a.



C1		Right				
		you	like	football	and	…
Left	you	0	1	0	1	…
	like	0	0	1	0	…
	football	0	0	0	0	…
	and	0	0	0	0	…
	…	…	…	…	…	…

C1	you	like	football	and	…
you	0	1	0	1	…
…	…	0	…	…	…

Fig. 68: Functioning scheme of the trained NLP algorithm

As shown in Fig. 68, the input sentence is first encoded according to the enhanced BOW model by means of Eqs. (18) and (19). The context understanding (cf. Fig. 61a) is then realized by extracting the LTR model indicated by the ANN output from the global database. Clearly, the selected LTR model (C1) corresponds to the highest probability resulted from the ANN (y_1, cf. Fig. 68). As previously illustrated in Fig. 66b, the sentence continuation (cf. Fig. 61a) is finally accomplished by feeding the last word of the input sentence (“you”) into the selected LTR model and obtaining the possible continuations “like” and “and” (cf. Fig. 68). The output sentences “do you like” and “do you and” can be potentially re-fed into the NLP algorithm to be further continued.

3.1.2. Testing of the developed NLP algorithm

In this section, the NLP algorithm (cf. Fig. 68) is trained by adopting an initial training set which is much larger than that reported in Table 64, enabling the reader to become more familiar with some critical technical aspects of the training process. The trained NLP algorithm is finally tested in order to verify its robustness and to highlight its potential limits regarding our purposes (cf. Table 9). The considered training set is composed by 50 different sentences belonging to the database of the most commonly used words[21] in the Corpus of Contemporary American English[22] (COCA) [116], and it is reported for brevity in annex A.1. Table 72 shows a reduced section of the training set and of the corresponding enhanced BOW model (cf. Table 64), where the 133 words of the total lexicon are reported in descending order of frequency.

Table 72: Reduced section of the considered training set (cf. annex A.1) and of the corresponding enhanced BOW model.

#	Sentence	Enhanced BOW (total lexicon = 133 words)							
		I	is	you	my	this	to	…	year
S1	will you be my friend	0	0	0.3	0.7	0	0	…	0
S2	you and I will always be friends	0.36	0	0.07	0	0	0	…	0
S3	today is the first of november	0	0.25	0	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…
S48	there are so many things I want to learn	0.61	0	0	0	0	0.83	…	0
S49	this is the year I am going to learn english	0.45	0.15	0	0	0.05	0.75	…	0.35
S50	I am so sorry	0.13	0	0	0	0	0	…	0

Presence Frequency	21	19	15	9	9	8	…	1

Notably, each sentence of the training set (cf. annex A.1) contains at least one word among the first 6 shown in Table 72 , i.e. the words “I”, “is”, “you”, “my”, “this” and “to”, which can be thus considered as the most representative features of the training set [111,114]. For this reason, these 6 features are adopted as the CFs for the clustering process in the pre-training step (cf. section 3.1.1.2). The current section is organized as follows. The two steps of pre-training and fine-training (cf. Fig. 62) are respectively described in sections 3.1.2.1 and 3.1.2.2. The results provided by the trained NLP algorithm and its limits are finally discussed in section 3.1.2.3.

3.1.2.1. Pre-training: clustering and construction of the LTR database

According to the 6 CFs adopted (“I”, “is”, “you”, “my”, “this” and “to”), the training dataset to cluster corresponds to the first 6 columns of the enhanced BOW model reported in Table 72, and it thus consists of 50 samples defined in R^6. Based on the tests presented in section 2.2.3.6, the clustering procedure was conducted with the implemented GA by testing seven different numbers of clusters, from K=2 to K=8. For brevity, the parameters used in the GA (cf. Fig. 56) are summarised in annex A.2 for

[21] Find the training sentences at https://gonaturalenglish.com/1000-most-common-words-in-the-english-language/.

[22] For more details, see https://www.english-corpora.org/coca/.

each K. The resulting values of D^K, R_D and ΔR_D are reported in Table 73 (cf. Eq.(15) and Table 44), while the corresponding curve K-R_D is shown in Fig. 69.

Table 73: Values of D^K, R_D and ΔR_D.

K	D^K	R_D [%]	ΔR_D [%]
1	13.27	0	---
2	10.48	21	21
3	8.21	38	17
4	5.97	55	17
5	4.39	67	12
6	3.61	73	6
7	2.95	78	5
8	2.68	80	2

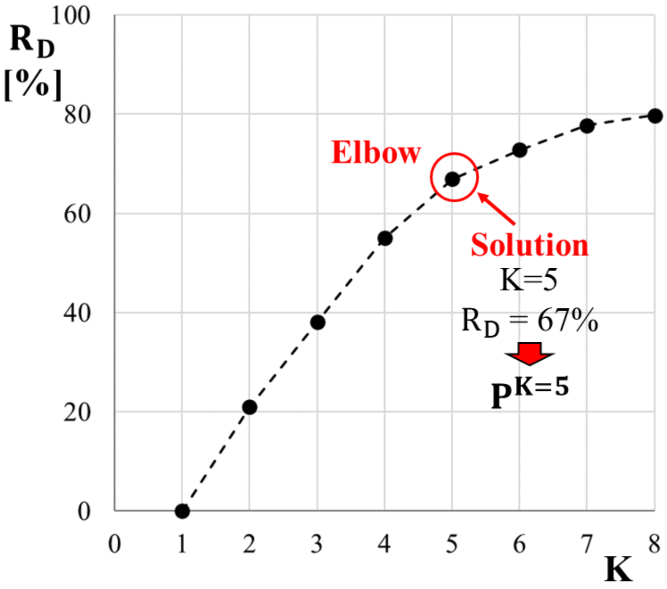


Fig. 69: Curve K-R_D.

The elbow of the curve K-R_D can be identified with the help of ΔR_D, whose value is halved from 12% to 6% when passing from K=5 to K=6 (cf. Table 73). The partitioning $P^{K=5}$ = {C1, C2, C3, C4, C5} is thus chosen as solution of the clustering procedure (cf. Fig. 69). The training samples contained in the clusters and the features of the respective ASs are provided in Table 74. Each AS is characterised by 6 features, which correspond to the 6 CFs adopted ("I", "is", "you", "my", "this" and "to").

Table 74: Content of the clusters and features of the respective ASs in the partitioning $P^{K=5}$ (cf. Fig. 69).

			ASs' features (CFs)					
Cluster	Training samples (sentences) #	AS	I	is	you	my	this	to
C1	9, 11, 21, 23, 29, 31, 34, 40, 42, 43	AS1	0	0.53	0	0	0.27	0
C2	12, 15, 17, 32, 35, 44	AS2	0.13	0.13	0.73	0	0	0.13
C3	6, 22, 33, 41, 47, 48, 49	AS3	0.27	0.07	0	0	0	0.67
C4	1, 14, 16, 19, 24, 36, 46	AS4	0	0.13	0.07	0.67	0	0
C5	2, 3, 4, 5, 7, 8, 10, 13, 18, 20, 25, 26, 27, 28, 30, 37, 38, 39, 45, 50	AS5	0.2	0.07	0.07	0	0	0

We can see that the training samples are not uniformly distributed in the five clusters. According to Table 74, C5 contains the 40% (20/50) of all the training samples, while the other clusters contain far fewer samples, i.e. between the 12% (C2) and the 20% (C1). This because AS5 is characterised by low values of the features "I", "is" and "you", which often appear among the first words of the considered training sentences (cf. annex A.1). In the fine-training step (cf. Fig. 62), the non-uniform distribution of the training samples in the obtained clusters would lead to generate many calibration samples belonging to the context C5 (cf. Fig. 67a), and far fewer samples belonging to the other contexts. This would involve a concrete risk of biasing the calibration process towards C5, since the ANN would achieve a deep knowledge of this context, but it would have a poor accuracy in the contexts different from C5 [111,114]. To avoid this problem, the number of samples in the clusters is uniformed by following the process illustrated in Fig. 70. The subscript "e" indicates that the clusters, the ASs and the CFs involved in Fig. 70 are taken as examples to facilitate the comprehension of the process, and they are thus different from those reported in Table 74. The number of samples of each cluster is indicated in the legend.

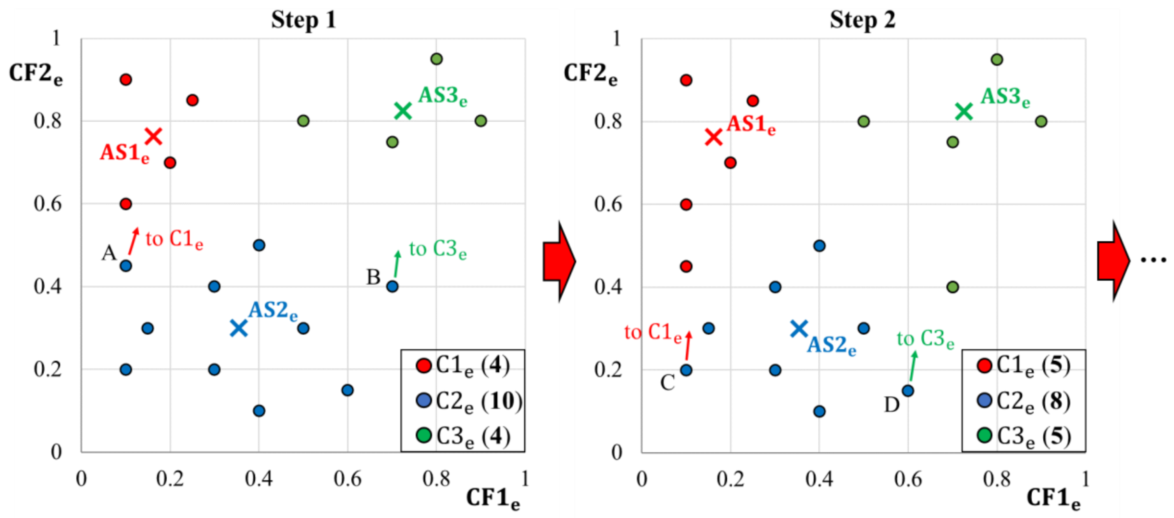


Fig. 70: Redistribution process of the outest samples into the nearest clusters to uniform the number of samples in the clusters.

At the beginning (step 1, cf. Fig. 70), C2$_e$ contains far more samples than C1$_e$ and C3$_e$, similarly to C5 in Table 74. Step by step, the outest samples in C2$_e$ are transferred into the respective nearest clusters, until the number of samples in all clusters is uniform. For each one of the outest samples to transfer, the destination cluster is determined by evaluating the Euclidean distance (cf. Eq. (13)) between the same sample and the ASs of the other clusters. In step 1, the outest samples A and B are respectively transferred into C1$_e$ and C3$_e$, thus reducing by two the number of samples in C2$_e$ and increasing by one those of C1$_e$ and C3$_e$. The same effect is obtained in step 2, where the outest samples C and D are transferred into C1$_e$ and C3$_e$, respectively (cf. Fig. 70). At the end, the clusters C1$_e$, C2$_e$ and C3$_e$ will be characterised by six samples each one. It is important to remark that the process illustrated in Fig. 70 enables to uniformly redistribute the training samples without changing the cores of the clusters obtained with the GA. Moreover, it can be easily adapted to our training set in R^6 (first six columns of Table 72) to uniform the clusters in Table 74. The resulting uniformed clusters are reported in Table 75 and they contain ten samples each one. Observing the old clusters in Table 74, we can see that C1 has not been modified by the redistribution process, while the first ten samples of the previous C5 have been redistributed in C2, C3 and C4. After completing the samples' redistribution process (cf. Fig. 70), the LTR models of the uniformed clusters (cf. Table 75) are constructed by following the process in Fig. 65, and they are finally organized in the global three-dimensional LTR database represented in Fig. 71 (cf. Fig. 66b).

Table 75: Uniformed clusters of the partitioning P$^{K=5}$.

Cluster	Training samples (sentences) #
C1	9, 11, 21, 23, 29, 31, 34, 40, 42, 43
C2	1, 7, 12, 13, 15, 17, 20, 32, 35, 44
C3	2, 6, 8, 18, 22, 33, 41, 47, 48, 49
C4	3, 4, 5, 10, 14, 16, 19, 24, 36, 46
C5	25, 26, 27, 28, 30, 37, 38, 39, 45, 50

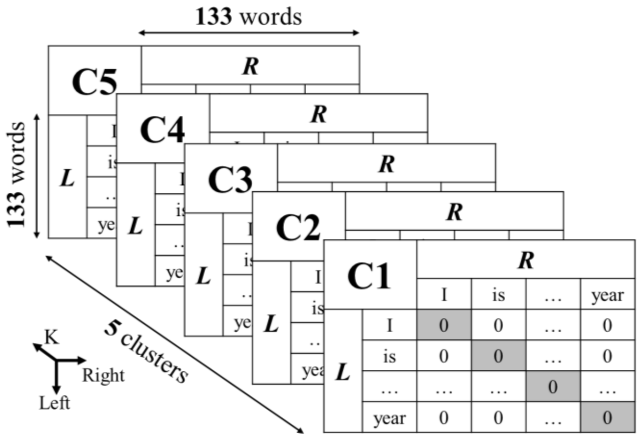


Fig. 71: Global LTR database of the partitioning P$^{K=5}$ (cf. Table 74).

According to the obtained partitioning ($P^{K=5}$, cf. Table 75), the global LTR database is composed by five LTR models, whose rows and columns contain the 133 words of the total training lexicon (cf. Table 72). Thanks to the redistribution process of the samples, the five LTR model share a similar amount of lexical and syntactic knowledge, since the number of training sentences in the starting clusters is the same (cf. Table 75).

3.1.2.2. Fine-training: generation of the calibration set and ANN calibration

The calibration set is generated with the process in Fig. 67a, and it consists of 2000 calibration samples equally divided into the five output contexts (cf. Table 75). This enables to observe the calibration guide principles reported in Table 35 in terms of number and diversity of calibration samples. A reduced section of the calibration set is reported in Table 76. Similarly to the set in Table 70, the current calibration set includes both mask-free and mask-with sentences. According to Table 76, the ANN to be calibrated is characterised by 133 input neurons, one for each word of the total lexicon, and 5 output neurons, one for each context (cf. Fig. 67b), while the number of hidden neurons is determined during the calibration.

Table 76: Calibration set generated with the process in Fig. 67a.

	Calibration input						**Calibration output**				
Sample #	*Incomplete sentence*	*Orig. sent.*	*Enhanced BOW (133 words)*				*Context (cluster)*	*Probability*			
			I	**is**	**…**	**year**		**t_1**	**t_2**	**…**	**t_5**
1	MASK is the first	S3	0	0.38	…	0	C5	0	0	…	1
2	I want to go with	S15	0.1	0	…	0	C2	0	1	…	0
3	this is MASK year	S49	0	0.38	…	0.88	C3	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…
2000	that MASK is	S11	0	0.83	…	0	C1	1	0	…	0

Based on the tests presented in section 2.2.2.6, six calibration tests with increasing numbers of hidden neurons (5, 10, 15, 20, 25, 30) were conducted with the implemented ANN. Each test refers to a different calibrated ANN, whose accuracy is evaluated as function of E^{v}_{ep} (cf. section 2.2.2.6 and Table 37). For brevity, the used set-up parameters of the ANN (cf. Fig. 39) are reported in annex A.3. The values of E^{v}_{ep} achieved in the calibration tests are reported in Table 77, together with the values of ΔE^{v}_{ep} (cf. Table 37).

Table 77: Values of E^{v}_{ep} obtained in the calibration tests and corresponding percentage variations (ΔE^{v}_{ep}).

Test # (ANN #)	**Number of hidden neurons**	**E^{v}_{ep}**	**ΔE^{v}_{ep} [%]**
1	5	0.1	---
2	10	0.1	0
3	15	0.1	0
4	20	0.1	0
5	25	0.1	0
6	30	0.1	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 90%

We can easily remark that the same values of E^{v}_{ep} were obtained in all tests, indicating an achieved accuracy of 90% for all the calibrated ANNs. Based on section 2.2.2.6, the calibrated ANN with the minimum number of hidden neurons (5, cf. Table 77) is thus adopted for the NLP algorithm (cf. Fig. 68).

3.1.2.3. Results and limits of the trained NLP algorithm

Based on Fig. 68, the NLP algorithm schematized in Fig. 72a was implemented in Matlab, and it involves the calibrated ANN adopted (10 hidden neurons and 90% accuracy) and the LTR database constructed in Fig. 71. Three examples showing the results of sentence continuation are reported in Fig. 72b, where the superscript of the words indicates the iteration number. At each iteration, the output sentences obtained at the previous iteration were entered into the NLP algorithm, until the suggested continuations were exhausted. For instance, the sentence "can you give my favorite" was obtained in the first iteration with the initial input "can you give me my" (cf. Fig. 72a), while the sentence "can you give my favorite cookie" resulted from the second iteration by considering the sentence "can you give my favorite" as input. Each example is characterised by a different tree structure, depending on which context of the global LTR database was involved in sentence continuation.

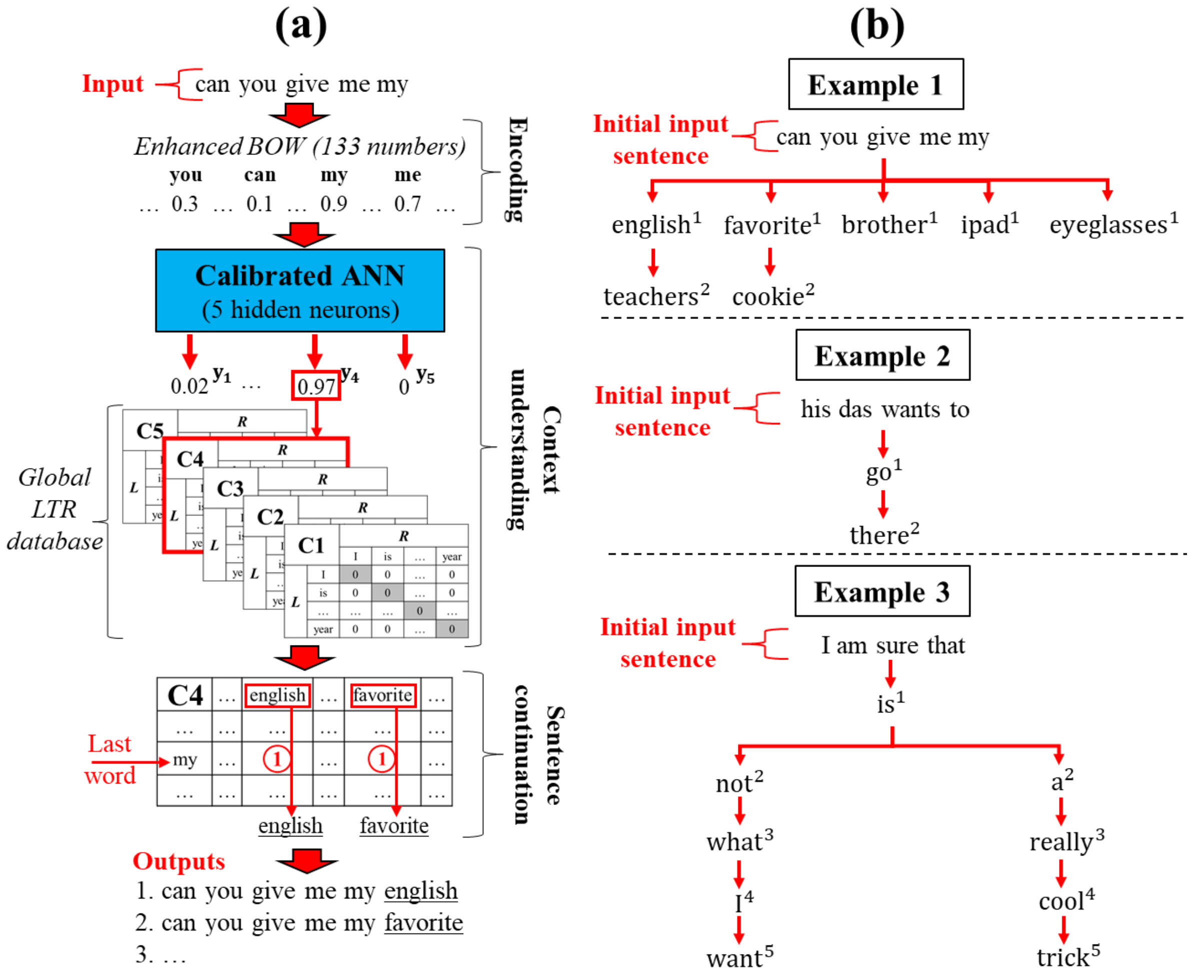


Fig. 72: (a) Scheme of the trained NLP algorithm (cf. Fig. 68); (b) examples of sentence continuation.

All three initial input sentences are different from the original training sentences and contain unknown words, i.e. "give", "wants" and "sure" respectively, which are not included in the lexicon of the original training set (cf. annex A.1). In general, the NLP algorithm is quite robust in processing unknown sentences and in proposing different coherent continuations, thanks to the use of masked words in the calibration set of the ANN (cf. Table 71). This is a very positive result, mostly considering that the initial training set (50 sentences and 133 different words) is still very limited compared to that of a real person, which already includes about 6,000 different words in 6-years-old children [117]. We can remark that quite singular and unexpected output sentences are obtained in some cases, as "can you give me my english teachers" and "can you give me my brother" in example 1 (cf. Fig. 72b). In accordance with our purposes (cf. Table 9), obtaining unexpected and singular results represents an advantage in view of conceptual design, since solutions that seem uncoherent or even meaningless enable to further enlarge the solutions space and could hide a great breakthrough potential. Based on Fig. 61a and on our purposes (cf. Table 9), some limits of the developed NLP methodology must be however highlighted. The number of possible continuations proposed clearly depends on the adopted training set (cf. Table 72), and on how and how many times the different words of the training lexicon are employed. Therefore, very few possible continuations can be often suggested by the NLP algorithm, mostly in the first iterations (as in the examples 2 and 3, cf. Fig. 72b). The number of continuations proposed at each iteration can be directly increased by adding new sentences to the initial training set (cf. Table 72). This however requires re-training the NLP algorithm, and it may thus be very time consuming. An alternative approach consists of selecting more contexts during the phase of context understanding and to combine them, in order to increase the effective number of proposed continuations. This enables to exploit the current training set as much as possible without adding new training sentences. The latter approach is described in section 3.1.3, and it represents an improvement of the current NLP algorithm.

3.1.3. Improvement of the NLP methodology

The improvement of the NLP methodology described in this section enables to increase the number of sentence continuations proposed by the NLP algorithm without considering additional training sentences. The same training set of 50 sentences as Table 72 (cf. annex A.1) is assumed in order to compare the final results to those obtained in Fig. 72b. According to [111,114], the 6 most frequent words, i.e. "I", "is", "you", "my", "this" and "to", were considered as the most representative features and were thus adopted as CFs in section 3.1.2. In this section, the idea is to divide the same CFs into two distinct groups, rather than considering them as a unique ensemble. This will enable to obtain two distinct LTR databases to be combined in the phase of context understanding (cf. Fig. 72b), increasing the number of proposed continuations. The current section is organized as follows. The training process (cf. Fig. 62) is described in section 3.2.3.1. The scheme of the improved NLP algorithm is presented in section 3.2.3.2. Results and conclusions are finally provided in section 3.2.3.3.

3.1.3.1. Training process

Let us divide the CFs into two groups: CFs^1 = {"I", "is", "you"} and CFs^2 = {"my", "this", "to"}. The respective datasets to cluster are reported in Table 78 (cf. Table 72), and they both consist of 50 samples defined in R^3.

Table 78: Training datasets to cluster according to the two adopted groups of CFs (cf. Table 72).

	Dataset 1 (CFs^1)			Dataset 2 (CFs^2)		
Sample #	**I**	**is**	**you**	**my**	**this**	**to**
S1	0	0	0.3	0.7	0	0
S2	0.36	0	0.07	0	0	0
S3	0	0.25	0	0	0	0
…	…	…	…	…	…	…
S48	0.61	0	0	0	0	0.83
S49	0.45	0.15	0	0	0.05	0.75
S50	0.13	0	0	0	0	0

The steps of pre-training and fine-training are conducted in the same way as in sections 3.1.2.1 and 3.1.2.2, respectively. The only difference is that we will construct two distinct LTR databases, one for each dataset (cf. Table 78) and we will calibrate two different ANNs, one for each LTR database. The pre-training and fine-training steps are briefly presented as follows.

Pre-training. As in section 3.1.2.1, two different clustering procedures, one for each training datasets (cf. Table 78), were conducted with the implemented GA, by testing seven different numbers of clusters, from K=2 to K=8. The set-up parameters of the GA are the same as annex A.2, while the curves K-R_D and the resulting partitioning levels inherent to CFs^1 and CFs^2 are reported in annex A.4 for brevity. The partitioning levels inherent to CFs^1 and CFs^2 are both composed by 4 clusters (K=4), and they are respectively indicated as $P_1^{K=4}$ = {C1.1, C1.2, C1.3, C1.4} and $P_2^{K=4}$ = {C2.1, C2.2, C2.3, C2.4}. For $P_1^{K=4}$ and $P_2^{K=4}$, the uniformed clusters are obtained with the redistribution process in Fig. 70 , and they are reported in Table 79 and Table 80, respectively. As in section 3.1.2.1, this avoids biasing the calibration of the ANN during the step of fine-training.

Table 79: Uniformed clusters of the partitioning $P_1^{K=4}$ (CFs^1).

Cluster	Training samples (sentences) #
C1.1	6, 16, 19, 22, 24, 33, 36, 40, 42, 45, 46, 50
C1.2	2, 3, 4, 14, 21, 23, 25, 26, 27, 28, 35, 41, 43
C1.3	1, 5, 15, 29, 30, 31, 32, 34, 37, 38, 39, 44
C1.4	7, 8, 9, 10, 11, 12, 13, 17, 18, 20, 47, 48, 49

Table 80: Uniformed clusters of the partitioning $P_2^{K=4}$ (CFs^2).

Cluster	Training samples (sentences) #
C2.1	9, 10, 11, 13, 14, 16, 17, 18, 19, 24, 36, 46
C2.2	28, 29, 30, 31, 32, 34, 35, 37, 38, 39, 44, 45, 50
C2.3	3, 4, 6, 15, 20, 21, 22, 25, 33, 41, 47, 48, 49
C2.4	1, 2, 5, 7, 8, 12, 23, 26, 27, 40, 42, 43

The LTR databases inherent to $P_1^{K=4}$ and $P_2^{K=4}$ are respectively represented in Fig. 73a and b (cf. Fig. 71). Each LTR model was constructed with the training sentences contained in the corresponding uniformed cluster (cf. Table 79 and Table 80) by following the process in Fig. 65, and its rows and columns contain the 133 words of the total training lexicon (cf. Table 72).

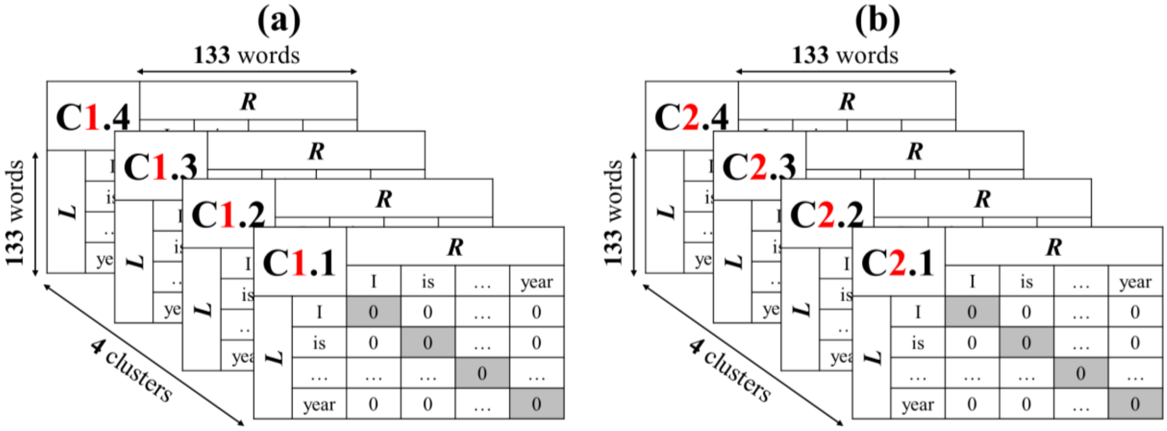


Fig. 73: (a) LTR database of the partitioning $P_1^{K=4}$ (CFs^1) and (b) LTR database of the partitioning $P_2^{K=4}$ (CFs^2).

Fine-training. The calibration set consists of 2000 different samples and it is reported in Table 81. Notably, the same calibration inputs as those in Table 76 were used to allow a more consistent comparison between the improved NLP algorithm and the NLP algorithm represented in Fig. 72b. As shown in Table 81, we have two different sets of calibration outputs, the first one inherent to $P_1^{K=4}$ (cf. Table 79) and to the LTR database in Fig. 73a and second one inherent to $P_2^{K=4}$ (cf. Table 80) and to the LTR database in Fig. 73b. Two different ANNs must be thus calibrated, one for each set of calibration outputs and both characterised by 133 input neurons and 4 output neurons (cf. Table 81), while the respective numbers of hidden neurons are determined during the calibration process.

Table 81: Calibration set (cf. Table 76).

	Calibration input					**Cal. output 1 ($P_1^{K=4}$)**				**Cal. output 2 ($P_2^{K=4}$)**			
S. #	*Incomplete sentence*	*Orig. sent.*	*Enhanced BOW (133 words)*			*Cont.*	*Probability*			*Cont.*	*Probability*		
			I	**…**	**year**		**t_1**	**…**	**t_4**		**t_1**	**…**	**t_4**
1	MASK is the first	S3	0	…	0	C1.2	0	…	0	C2.3	0	…	0
2	I want to go with	S15	0.1	…	0	C1.3	0	…	0	C2.3	0	…	0
3	this is MASK year	S49	0	…	0	C1.4	0	…	1	C2.3	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
2000	that MASK is	S11	0	…	0	C1.4	0	…	1	C2.1	1	…	0

For both sets of calibration outputs (cf. Table 81), different calibration tests with increasing numbers of hidden neurons were conducted with the implemented ANN, similarly to section 3.1.2.2 and using the set set-up parameters reported in annex A.3. For brevity, the results of the calibration tests in terms of E_{ep}^{v} and the choice of the calibrated ANNs are summarised in annex A.5. The characteristics of the calibrated ANNs adopted for the improved NLP algorithm are reported in Table 82, in terms of number of hidden neurons, corresponding E_{ep}^{v} and average accuracy. We can see that ANN1 and ANN2 are respectively characterised by 15 and 20 hidden neurons. Their values of E_{ep}^{v} are quite similar (0.08 and 0.06, respectively), indicating an average accuracy of 93%.

Table 82: Characteristics of the calibrated ANNs adopted for the improved NLP algorithm (cf. annex A.5).

Calibrated ANN	**Corresponding partitioning**	**Number of hidden neurons**	**E_{ep}^{v}**	**Average accuracy**
ANN1	$P_1^{K=4}$	15	0.08	93%
ANN2	$P_2^{K=4}$	20	0.06	

3.1.3.2. Scheme of the improved NLP algorithm

The scheme of the improved NLP algorithm is illustrated in Fig. 74, involving the two ANNs calibrated in the fine-training step (cf. Table 82) and the two LTR databases constructed in the pre-training step (cf. Fig. 73). The same input sentence considered in Fig. 72a is initially encoded into the same enhanced BOW composed by 133 numbers, which is now fed into both ANN1 and ANN2. During the phase of context understanding, two LTR models are extracted from the respective databases according to the outputs of the respective ANNs. The selected LTR models are square matrices of the same size (133x133, cf. Fig. 73), and they can be thus easily combined into a unified context (C_U) by means of a weighted sum, as shown in Fig. 74. This operation not only increases the number of sentence continuations proposed by the NLP algorithm, but the values of C_U also represent a sort of fitness of the proposed continuation words to the input sentence. Clearly, the validity of the fitness depends on the initial training sentences. To better understand the results provided by the improved NLP algorithm, the same three examples presented in Fig. 72b will be resumed in section 3.1.3.3.

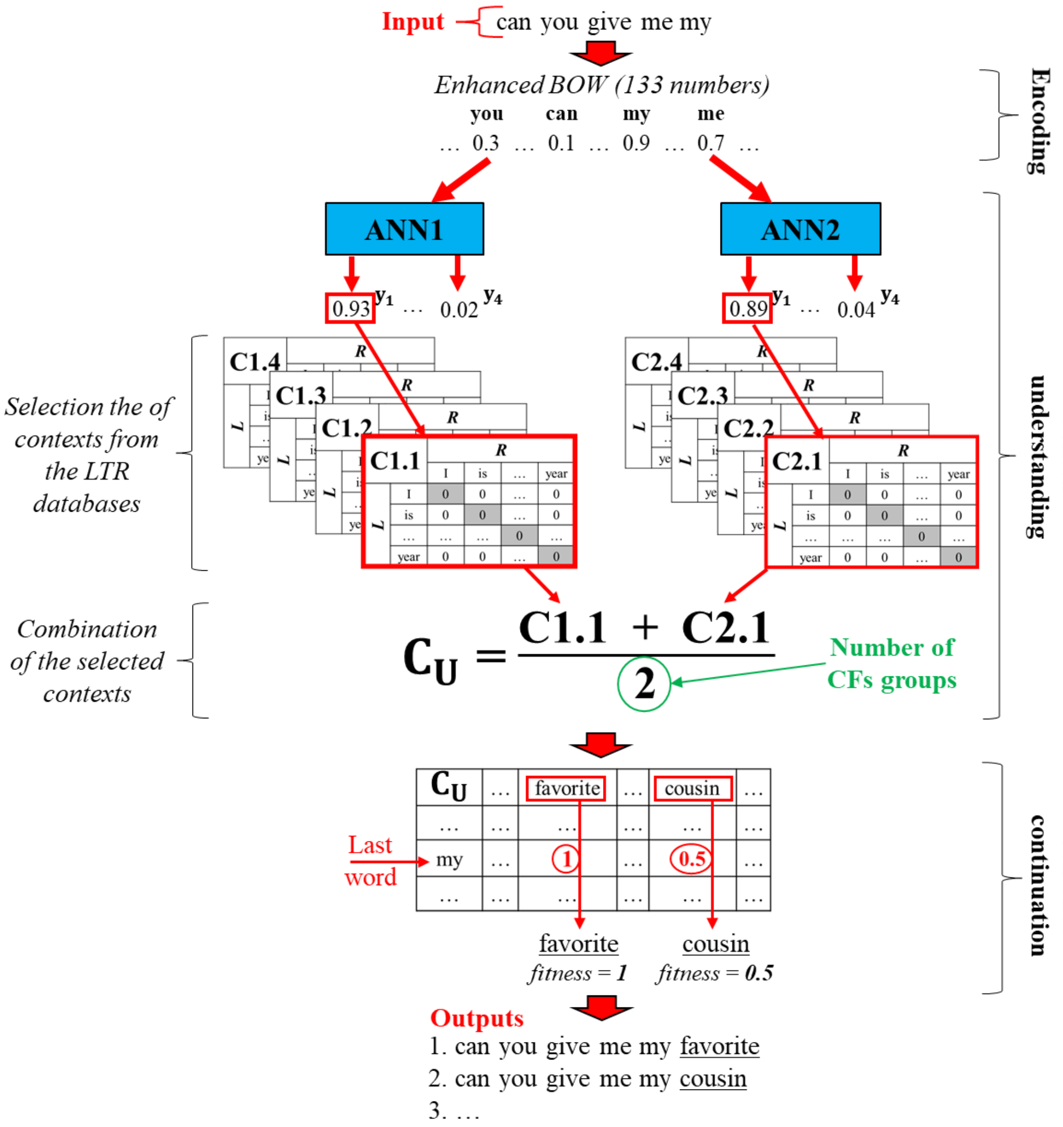


Fig. 74: Scheme of the improved NLP algorithm.

3.1.3.3. Results and conclusions

The results shown in Fig. 75a were obtained with the improved NLP algorithm (cf. Fig. 74), using the same initial input sentences of the three examples in Fig. 72b. The superscript indicates the iteration number, and the tree structures resulted by entering the output sentences obtained at the previous iteration into the improved NLP algorithm, until exhausting the proposed continuations (cf. section 3.1.2.3). For brevity, only the first two iterations are shown in Fig. 75a. We remember that all the initial input sentences contain unknown words which are not included in the lexicon of the training set, i.e. “give”, “dad” and “wants” (cf. annex A.1). The results reported in Fig. 75a lead to the same conclusions made in section 3.1.2.3: despite the low number of training sentences and the limited lexicon, the improved NLP algorithm demonstrates to be quite robust in processing unknown sentences and in proposing different coherent continuations. As in the previous version (cf. Fig. 72a), some unexpected or meaningless outputs are possible, like the sentences and “can you give me my cousin” (example 1) and “his dad wants to the first” (example 2, cf. Fig. 75a). However, in this case, the presence of this type of outputs can be potentially indicated to the user by a fitness value lower than 1[23].

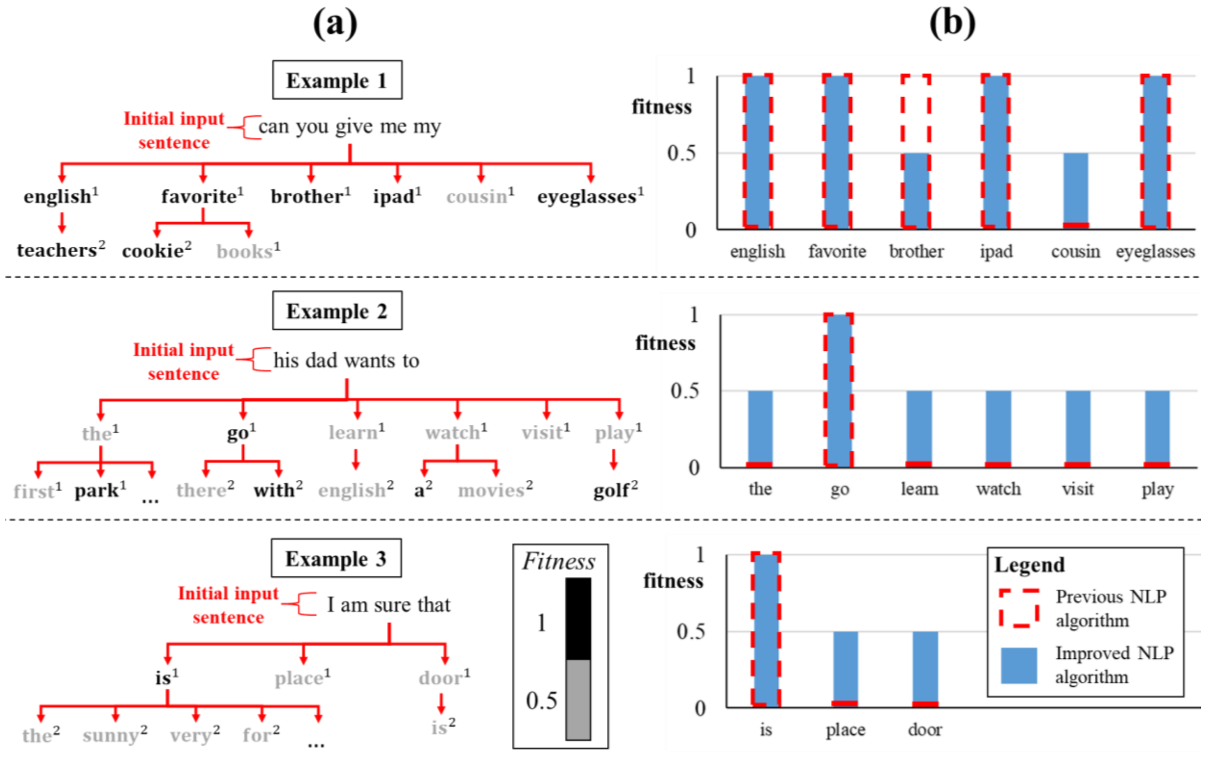


Fig. 75: (a) Results obtained with the improved NLP algorithm; (b) fitness histograms comparing the actual results to those in Fig. 72b at the first iteration.

The most significant result regards the number of proposed continuations, which was the purpose of the present section and which clearly is much higher than in the previous version. The fitness histograms in Fig. 75b compare the actual results to those reported in Fig. 72b at the first iteration of the NLP algorithm. For the previous NLP algorithm, the presence of a word among the proposed continuations is indicated by a red dotted bar of unitary length (cf. Fig. 75b). An increase of the number of proposed continuations can be remarked in all examples. Example 2 shows the most relevant increase, with 5 more words proposed at the first iteration than the previous version, and the 80% of them (“learn”, “watch”, “visit”, “play”) is clearly coherent with the context of the input sentence. We can conclude that the improved NLP algorithm fully satisfy our objective of increasing the number of proposed solutions without modifying the initial training set. Moreover, the presented approach can be further extended by adopting more than two distinct groups of CFs. Two important considerations can be finally added to the present discussion:

- The more distinct groups of CFs are adopted, the longer and the “more exhausting” the training process. Based on section 3.1.3.1, each additional group of CFs requires an additional clustering procedure to be performed, an additional LTR database to be constructed and an additional ANN to be calibrated. However, we believe the duration can be significantly reduced by including all these procedures in a unique automatic and autonomous training algorithm.

[23] In this version of the NLP algorithm, the fitness acts as indicator of potentially incoherent solution. In other words, a fitness lower than 1 does not necessarily indicate the corresponding solution is incoherent. Verifying the consistency of the obtained solutions lies with the user.

- The approach presented in this section can be considered as a short-term solution to improve the NLP algorithm potential, exploiting the training set as much as possible. In the long term, extending the operative range of the NLP algorithm clearly requires the enlargement of the training set. Nevertheless, we believe that the presented improvement is very useful when training data are difficult to find or when they take a long time to be collected, as we will see for our demonstrator.

The demonstrator is developed in the next section by adapting the methodology of the improved NLP algorithm to the field of conceptual design.

3.2. Demonstrator of conceptual design procedure based on the NLP methodology

The original demonstrator of conceptual design procedure presented in this section is based on the improved NLP algorithm (cf. section 3.1.3). The current section is organized as follows. The two main hypotheses on which the developed demonstrator is based are introduced and discussed in section 3.2.1. The adaptation of the improved NLP algorithm to conceptual design is presented in section 3.2.2. The initialization of the knowledge database and the training process of the NLP algorithm are presented in section 3.2.3. The functioning of the demonstrator and the obtained results are illustrated in section 3.2.4 with an original example of conceptual design. Some important future developments are finally proposed and discussed in section 3.2.5.

3.2.1. Hypotheses

We present the two main hypotheses of the developed demonstrator. The current section is organized as follows. The first hypothesis concerns the conceptual design procedure, and it is given in sections 3.2.1.1. The second hypothesis regards instead the knowledge base of the machine, and it is discussed in section 3.2.1.2. The hypotheses are finally summarised in section 3.2.1.3.

3.2.1.1. HP 1: Intervention of the user in the conceptual design procedure

In accordance with the PAAs reported in Table 18, the demonstrator of conceptual design procedure implemented in this work involves two main steps: identification of the SPs and development of the CVs (cf. Fig. 16). The scheme of the demonstrator represented in Fig. 76 is based on the original representation of the mental processes assumed in Fig. 28, and it highlights the roles of user and machine in performing the conceptual design steps. The description of the KCAs is reported in Table 24.

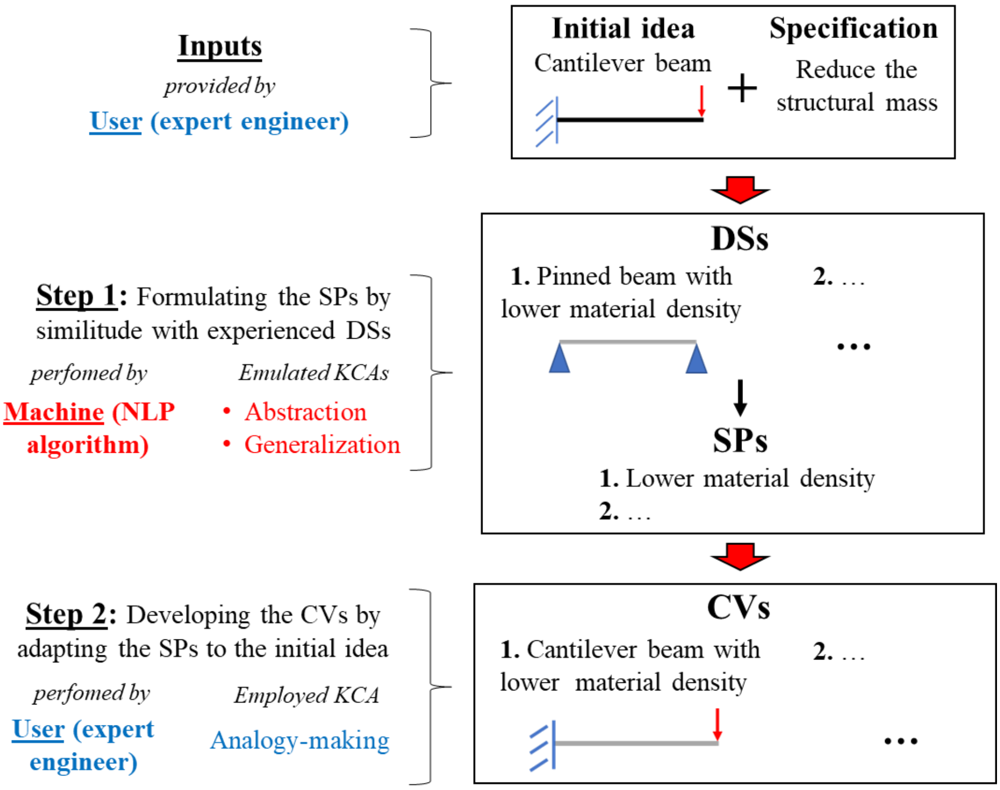

Inputs
provided by
User (expert engineer)
Initial idea
Cantilever beam
+
Specification
Reduce the structural mass
DSs
1. Pinned beam with lower material density
2. …
…
SPs
1. Lower material density
2. …
Step 1: Formulating the SPs by similitude with experienced DSs
perfomed by
Machine (NLP algorithm)
Emulated KCAs
• Abstraction
• Generalization
CVs
1. Cantilever beam with lower material density
2. …
…
Step 2: Developing the CVs by adapting the SPs to the initial idea
perfomed by
User (expert engineer)
Employed KCA
Analogy-making

Fig. 76: Scheme of the demonstrator highlighting the roles of user and machine in performing the conceptual design steps (cf. Fig. 28).

The input is initially provided by the user in form of initial idea and specifications. As highlighted in Fig. 76, it is assumed that the user is an expert in the field of mechanical and structural engineering. The first step is performed by the machine via the NLP algorithm, while the second step is performed by the user. As anticipated in section 2.1.2.3, the idea of the developed methodology based on machine learning is to exploit the power of the KCAs and all the advantages offered by machine during conceptual design, without being affected by brain limits (cf. Fig. 23). Due to the difficulties encountered in realizing this ambitious idea, our efforts in this work are mainly focused on the first conceptual design step. We are thus aware that the development of the CVs in the second step could be potentially influenced by memory biases and other brain limits (cf. Table 25). In the second step, it is also assumed to focus on the process of analogy-making for applying the SPs to the initial idea (cf. Fig. 28), while the new CVs will not be numerically evaluated. In agreement with our purposes (cf. Table 9), the idea is to remain at a conceptual level, diverting the user's attention from technical details concerning the mechanics of the developed CVs and which would affect the choices of the same user via the cognitive biases.

3.2.1.2. HP 2: Knowledge of the machine based on few words and few design samples

According to Fig. 76 and to the NLP methodology presented in section 3.1.3, the machine (NLP algorithm) exploits the knowledge of different design samples (DSs) to formulate the SPs by means of a certain number of words, whose ensemble constitutes the total lexicon. A limited lexicon (less than 30 words) and a limited number of DSs are assumed in the current demonstrator (cf. Fig. 76). Our position is clarified in Fig. 77, where the blue curve is adapted from [80,81,117] and it represents the evolution of the number of words known by a human as function of the age, while the red curve represents a hypothetic evolution of the lexicon known by the machine.

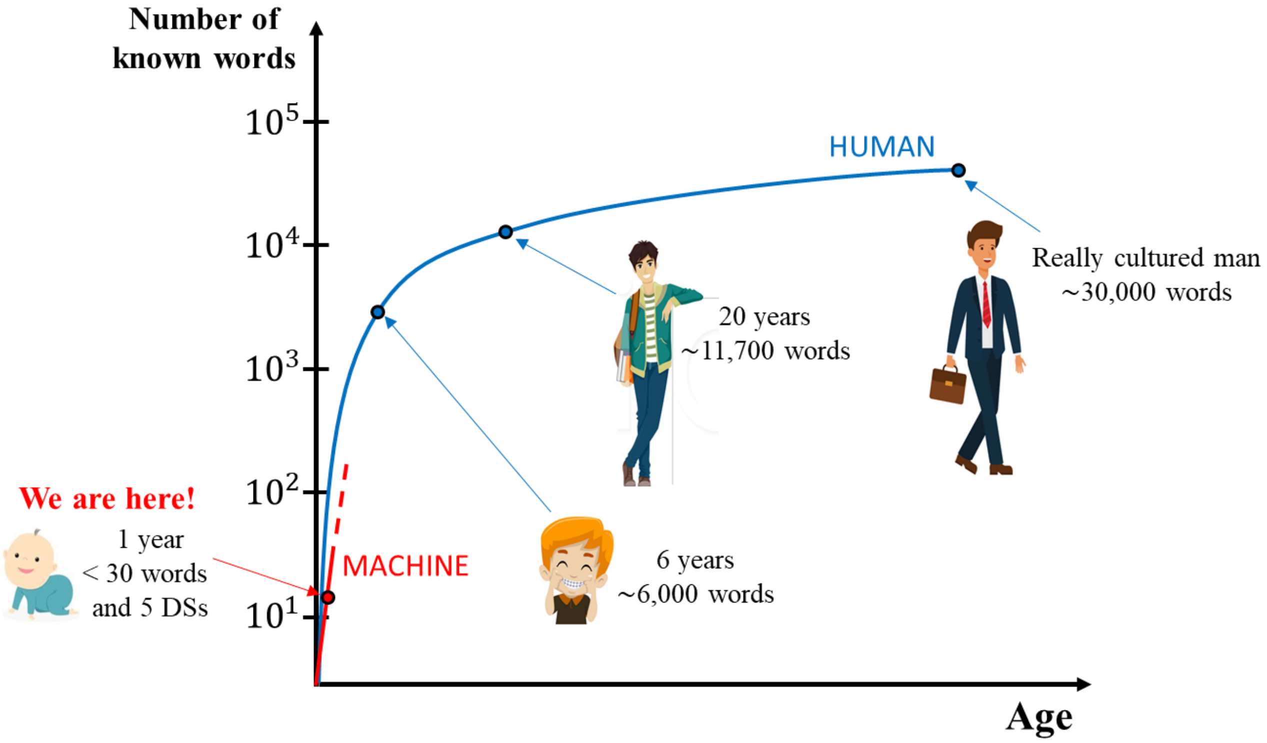


Fig. 77: Evolution of the known number of words (lexicon) in humans (cf. [80,81,117]) and in the machine of our demonstrator (cf. Fig. 76) as function of the age, highlighting the position of our work.

The lexicon of the machine and the known DSs are for now very limited, as in a baby learning his first words (cf. Fig. 77). The idea is to highlight the potential of the developed methodology by showing that the machine is able to express complex design concepts and to provide relevant results with few words and few design examples to which refer. The DSs and the adopted lexicon are presented as follows. The five considered DSs are shown in Fig. 78. All DSs involve bidimensional systems composed by one-dimensional structural elements with different behaviour models (wire, axial spring, truss and beam), as the standard tests for undergraduate engineering students. The behaviour models of all elements can be found in [42,118]. The assumptions made for the DSs are summarised in Table 83.

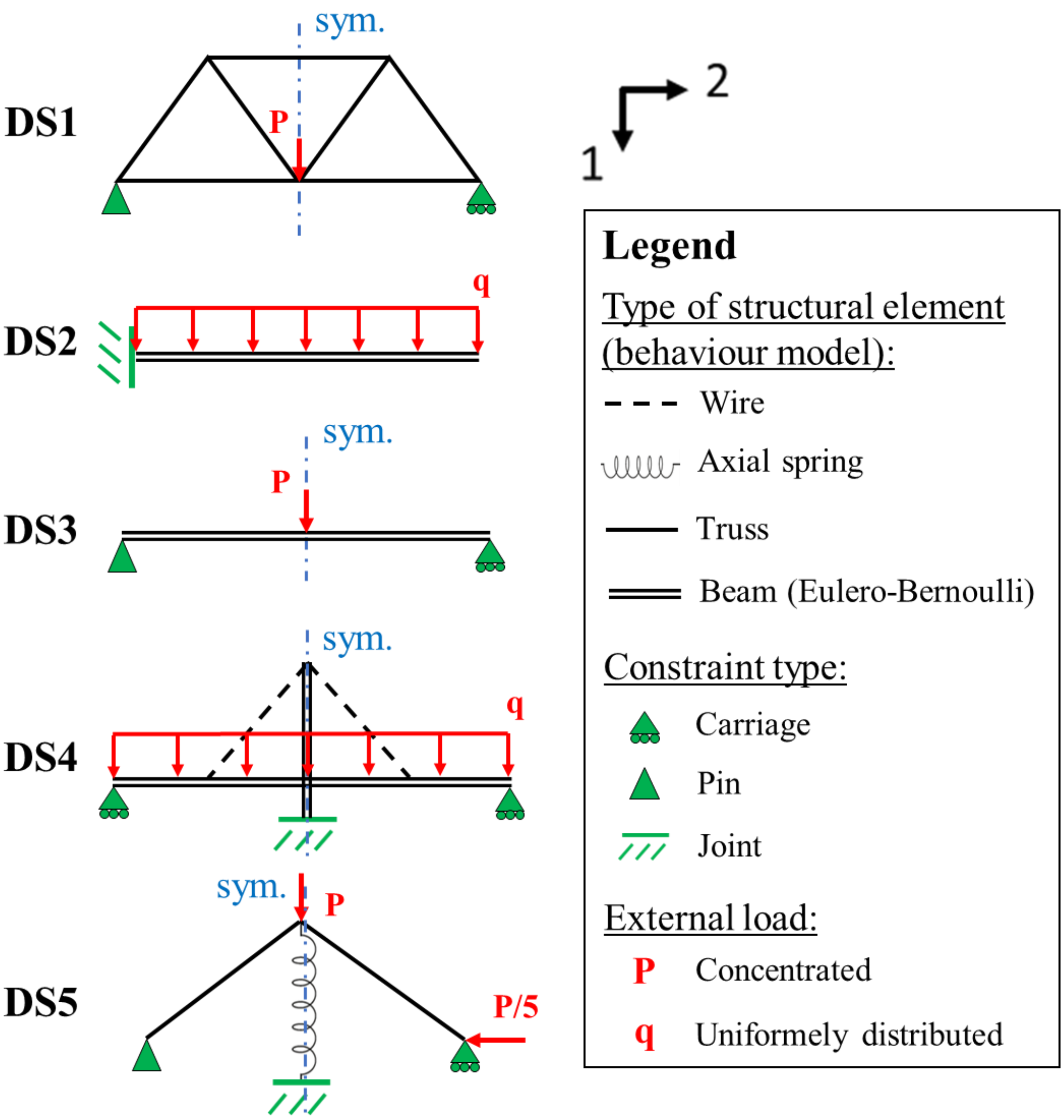


Fig. 78: Set of considered DSs.

Table 83: Assumptions on the considered DSs (cf. Fig. 78).

#	Assumption
1	Small displacement and small deformations
2	Constraints are perfect, smooth (no friction) and bilateral
3	Linear elastic isotropic and homogenous material, the spring is also linear
4	Generic section for beam elements

The adopted lexicon is composed by 28 different words (cf. Fig. 77) of three types:

- Clustering features (CFs), used to translate the DSs into BOW models, which will be employed to train the NLP algorithm (cf. Fig. 62). The 15 CFs assumed are reported in Table 84, and they are divided into two families and five distinct groups. According to the improved NLP methodology (cf. section 3.1.3), each group of CFs represents a different way of describing and clustering the DSs (cf. Table 78). The criteria to define the values of the CFs will be presented in section 3.2.2.

Table 84: Assumed CFs.

Family	Group #	Described aspect	CFs	Number of CFs
Spatial configuration	CFs^1	Geometry	free linear close	3
	CFs^2	Boundary conditions	straight extremity	2
Mechanical features	CFs^3	Element type	wire spring truss beam	4
	CFs^4	Constraint type	carriage pin joint	3
	CFs^5	Type of internal load	traction compression bending	3

- Engineering parameters (EPs), used to express the input specifications (cf. Fig. 76). Based on the GEPs in TIPS (cf. Table 12), the 6 EPs assumed are reported in Table 85, and they represent different generalized design specifications to potentially improve during conceptual design.

Table 85: Assumed EPs and respective descriptions.

EP	Description
m	Structural mass
U1	Absolute value of the maximum displacement along the direction 1 (cf. Fig. 78)
U2	Absolute value of the maximum displacement along the direction 2 (cf. Fig. 78)
Sr	Strength
Pk	Peak load of global buckling
Fn	First natural frequency

- Complementary lexicon (CL), used to express the SPs (cf. Fig. 76). The 7 words belonging this type are reported in Table 86, and they are of three types: material properties, section properties and structure topology. The use of the words regarding material and section properties is limited to the elements indicated in Table 86, in agreement with the assumptions of Table 83. The word “frame” enables to formulate SPs for directly acting on the structure topology, its use will be better clarified in section 3.2.4. It is important to remark that the words of the CL identify both parameters to be increased or decreased (as “E”, “rho”, etc.), and general characteristics to be added or removed (as “hollow”, “variable”, etc.) from the design of the initial idea. This will enable the machine to provide SPs of different types, in agreement with the BDPs and with our purposes (cf. Table 5 and Table 9).

Table 86: Words of the complementary lexicon (CL), respective description and use limitations.

Type	Word of the CL	Description	Use limited to
Material properties	E	Elastic modulus	Wire, spring, truss, beam
	rho	Density	Wire, truss, beam
Section properties	A	Area	Wire, truss, beam
	Jb	Bending inertia	Beam
	hollow	Beam with hollow section	Beam
	variable	Variable section over the length of the beam	Beam
Structure topology	frame	Different ensembles of connected structural elements	---

3.2.1.3. Summary of the hypotheses

The two hypotheses presented in this section are summarised in Table 87.

Table 87: Summary of the hypotheses.

HP	Assumptions
1. Intervention of the user in the conceptual design procedure (cf. section 3.2.1.1)	Scheme of the demonstrator presented in Fig. 76: • Inputs provided by the user in form of initial idea and specifications. The user is an expert of mechanical and structural engineering. • **Step 1** (identification of the SPs) performed by the machine via the NLP algorithm. • **Step 2** (development of the CVs) performed by the user (memory biases could be potentially introduced). Focus on analogy-making, CVs' numerical evaluation is not included.
2. Knowledge of the machine based on few words and few design samples (cf. section 3.2.1.2)	Very limited lexicon composed by 28 words: • 15 clustering features (CFs, cf. Table 84), • 6 engineering parameters (EPs) to express the specifications (EPs, cf. Table 85), • Complementary lexicon (CL) of 7 words to express the SPs (cf. Table 86). Only 5 DSs considered (cf. Fig. 78 and Table 83) The lexicon and the DSs assumed are sufficient to the machine for formulating complex design concepts and for providing relevant results.

3.2.2. Adaptation of the NLP methodology to conceptual design

The scheme of the NLP algorithm used in the first step of the demonstrator (cf. Fig. 76) is represented in Fig. 79, and it is based on the improved NLP algorithm illustrated in Fig. 74. The scheme enables to immediately visualize how the SPs are provided, and to better introduce the aspects discussed in this section. The explanation of Fig. 79 is conducted step by step as follows:

- **Inputs**. The initial idea is provided by the user as a structure to be improved. Following the ideas of the implemented NLP methodology, we adopted an arbitrary mechanical system different from the DSs in Fig. 78. The specifications are expressed by means of the EPs assumed in Table 85, where the use of the desinences "_+" (plus) or "_-" (minus) indicates whether the corresponding EP must be increased or decreased.
- **Encoding**. The initial idea is encoded in a BOW model composed by the 15 CFs assumed in Table 84. Similarly to the enhanced BOW used in section 3.1, all CFs are defined between 0 and 1. The criteria to define the values of the CFs are presented in the current section. As indicated in Fig. 79, this step emulates the KCA of abstraction (cf. Fig. 76): the CFs correspond to the abstracted features of the initial idea, and they will be used to retrieve the experienced DSs (cf. Table 24).
- **Context understanding and sentence continuation**. The encoded initial idea is fed into the ANNs of the NLP algorithm. As in Fig. 74, the number of ANNs and LTR databases involved in the context understanding corresponds to the number of CFs' groups (cf. Table 84). The rows and the columns of each LTR model respectively contain the specifications and the SPs. The latter are expressed by combining the words of the CL (cf. Table 86), the CFs (Table 84) and the desinences "_+" (plus), "_-" (minus) and "_w" (where). The definition and the interpretation of the SPs are described in the current section, while the construction of the LTR databases will be presented in section 3.2.3. As indicated in Fig. 79, these steps emulate the generalization ability (cf. Fig. 76): the conceptual solutions contained in the LTR databases are based on the DSs, but they are finally used as SPs on the initial idea (cf. Fig. 28).
- **Outputs**. According to our purposes (cf. Table 9), the resulting SPs involve conceptual solutions of different types, recommending both "quantitative" ("A_-") and "qualitative" ("hollow_+ bending_w") changes of the initial idea for reducing the mass.

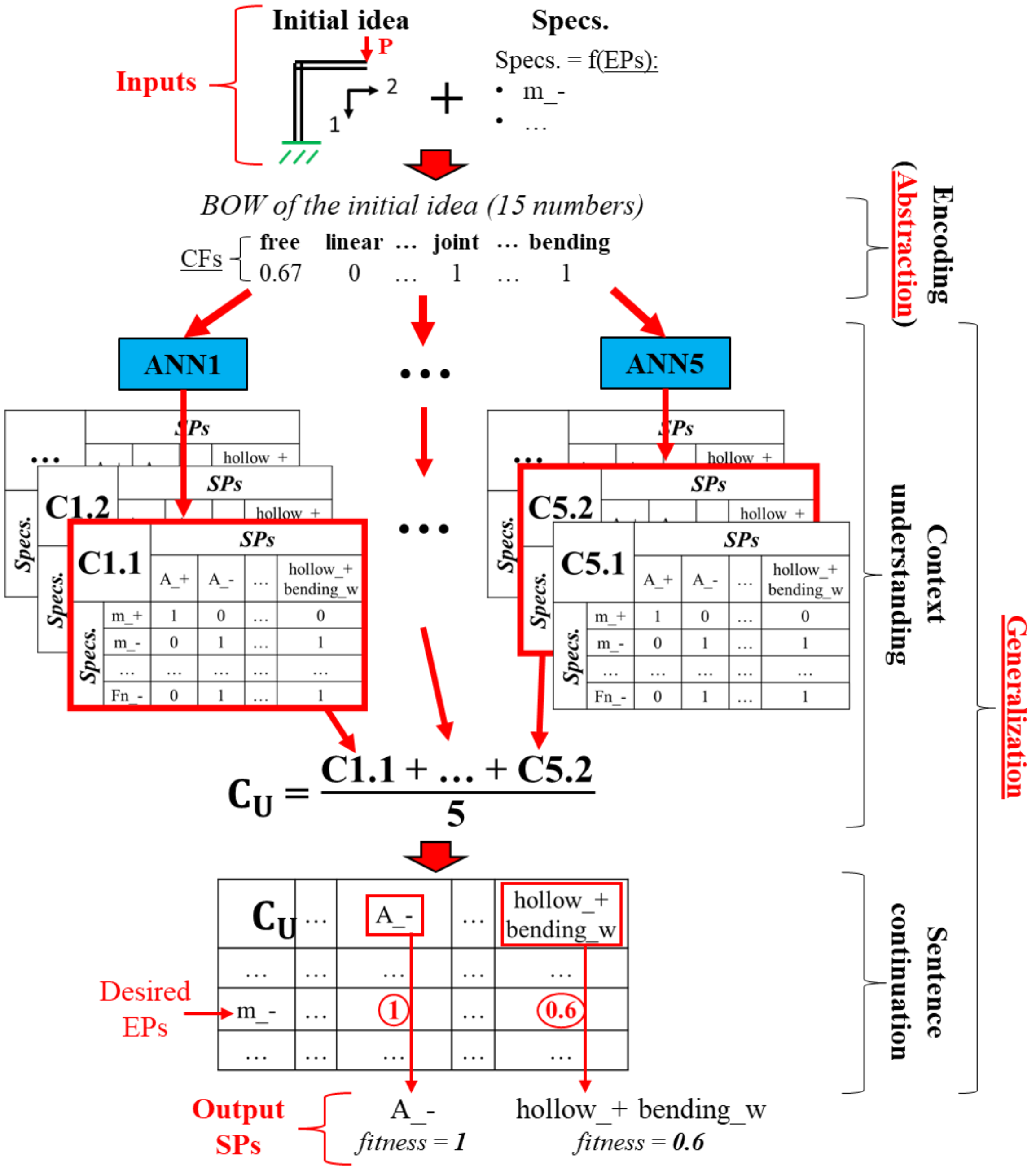


Fig. 79: Scheme of the NLP algorithm in the first step of the demonstrator (cf. Fig. 74 and Fig. 76).

The current section is organized as follows. The criteria to define the CFs' values are presented in section 3.2.2.1, and they represent an original and effective way of describing a mechanical structure with a natural language model inspired to the enhanced BOW (cf. section 3.1.1.1). All criteria were integrated in a useful tool implemented in MATLAB [48] for drawing the initial idea. This original tool directly provides the values of the CFs as function of the drawn structure, and it is briefly presented in section 3.2.2.2. The definition and the interpretation of the SPs are finally described in section 3.2.2.3.

3.2.2.1. Criteria to define the values of the CFs

According to Table 84, the CFs are divided into two families: spatial configuration and mechanical features. The CFs of the first family describe how the elements and the boundary conditions characterising the structure are arranged, without providing any information about the nature of the involved elements and constraints. Vice versa for the CFs of the second family, which also include the description about the nature of internal loads. The five structures shown Fig. 80 are used as examples to introduce the definition criteria of the CFs. All the structures are based on the legend in Fig. 78. We can see that S1 corresponds to the initial idea in Fig. 79, while DS3 and DS4 are among the design samples adopted in Fig. 78. The structures S1, S2 and S3 share a similar configuration of the elements, and they are used to better highlight the description role of the CFs.

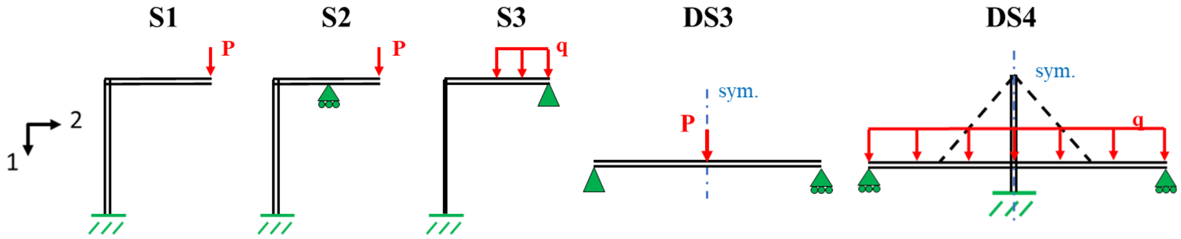
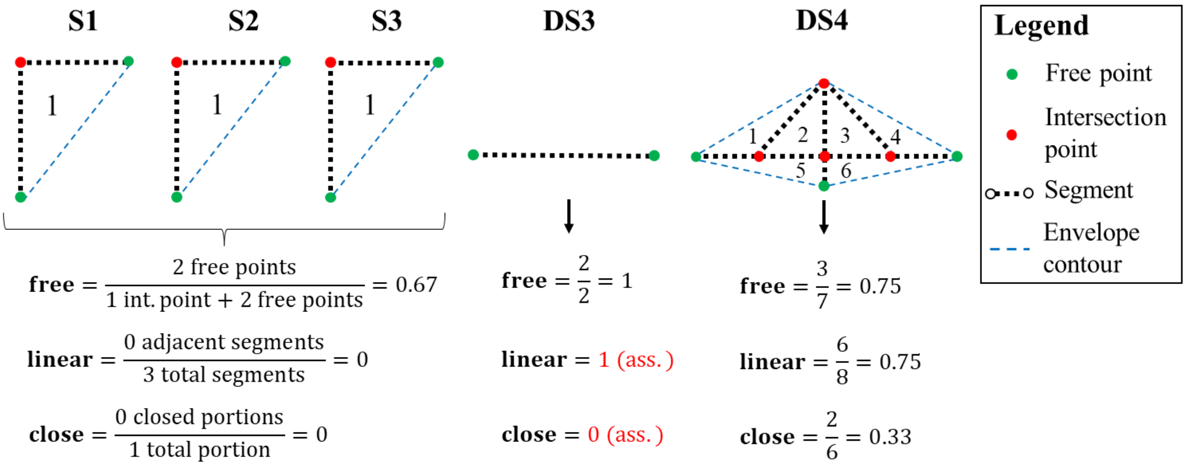


Fig. 80: Examples to introduce the definition criteria of the CFs, including three structures with a similar configuration and the design samples DS3 and DS4 (cf. Fig. 78).

The criteria are presented as follows for each group of CFs, according to the order in Table 84. The given criteria will be finally summarised in Table 88.

Group CFs^1 = {free, linear, close}. As shown in Fig. 81, the structures are merely considered as ensembles of segments delimited by points, neglecting all boundary conditions and all mechanical features. The red points in Fig. 81 correspond the intersections between the elements in the original structures (cf. Fig. 80), while the green points to the free vertices of the segments. The numbers on the structures of Fig. 81 correspond to the different portions identified after tracing the convex envelope contour, and they are assigned arbitrarily.



$$\textbf{free} = \frac{2\text{ free points}}{1\text{ int. point} + 2\text{ free points}} = 0.67 \qquad \textbf{free} = \frac{2}{2} = 1 \qquad \textbf{free} = \frac{3}{7} = 0.75$$

$$\textbf{linear} = \frac{0\text{ adjacent segments}}{3\text{ total segments}} = 0 \qquad \textbf{linear} = 1\text{ (ass.)} \qquad \textbf{linear} = \frac{6}{8} = 0.75$$

$$\textbf{close} = \frac{0\text{ closed portions}}{1\text{ total portion}} = 0 \qquad \textbf{close} = 0\text{ (ass.)} \qquad \textbf{close} = \frac{2}{6} = 0.33$$

Fig. 81: Definition criteria of the CFs in the first group (CFs^1 = {free, linear, close}, cf. Table 84).

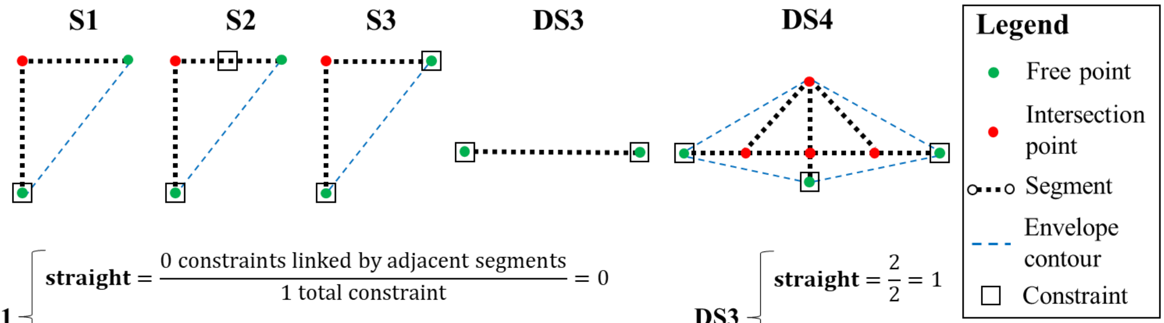
The three CFs belonging to this group are thus defined as follows:

- The feature "free" is the ratio of the number of free points to the sum of free points and intersection points (i.e. the total number of points).
- The feature "linear" is the ratio between the total number of adjacent segments (i.e. consecutive segments arranged on the same line) and the total number of segments. In case of a single segment (as DS3, cf. Fig. 81), the value of this feature is assumed as 1. In DS4, we have 6 total adjacent segments, i.e. the 4 horizontal ones plus the 2 vertical ones.

- The feature “close” is the ratio of the number of structurally closed portions (i.e. delimited by only segments) to the total number of portions. For example, the portions 2 and 3 in DS4 are structurally closed, while the portions 1, 4, 5 and 6 are structurally open. In case of a single segment (as DS3, cf. Fig. 81), the value of this feature is assumed as 0.

We can see that S1, S2 and S3 are characterised by the same values of the CFs “free”, “linear” and “close” the same values.

Group CFs^2 = {straight, extremity}. The representation of the structures considered in Fig. 81 is resumed in Fig. 82, where the positions of the constraints are now indicated in agreement with Fig. 80. Remark that we are considering the constraints as general boundary conditions by neglecting their nature.



$$\mathbf{S1}\begin{cases}\mathbf{straight} = \dfrac{\text{0 constraints linked by adjacent segments}}{\text{1 total constraint}} = 0\\[2ex] \mathbf{extremity} = \dfrac{\text{1 constraint coinc. with contour vertices}}{\text{1 total constraint}} = 1\end{cases}$$

$$\mathbf{DS3}\begin{cases}\mathbf{straight} = \dfrac{2}{2} = 1\\[2ex] \mathbf{extremity} = \dfrac{2}{2} = 1\end{cases}$$

$$\mathbf{S2}\begin{cases}\mathbf{straight} = \dfrac{0}{2} = 0\\[2ex] \mathbf{extremity} = \dfrac{1}{2} = 0.5\end{cases}$$

$$\mathbf{S3}\begin{cases}\mathbf{straight} = \dfrac{0}{2} = 0\\[2ex] \mathbf{extremity} = \dfrac{2}{2} = 1\end{cases}$$

$$\mathbf{DS4}\begin{cases}\mathbf{straight} = \dfrac{2}{3} = 0.67\\[2ex] \mathbf{extremity} = \dfrac{3}{3} = 1\end{cases}$$

Fig. 82: Definition criteria of the CFs in the second group (CFs^2 = {straight, extremity}, cf. Table 84).

According to Fig. 82, the two CFs belonging to this group are defined as follows:

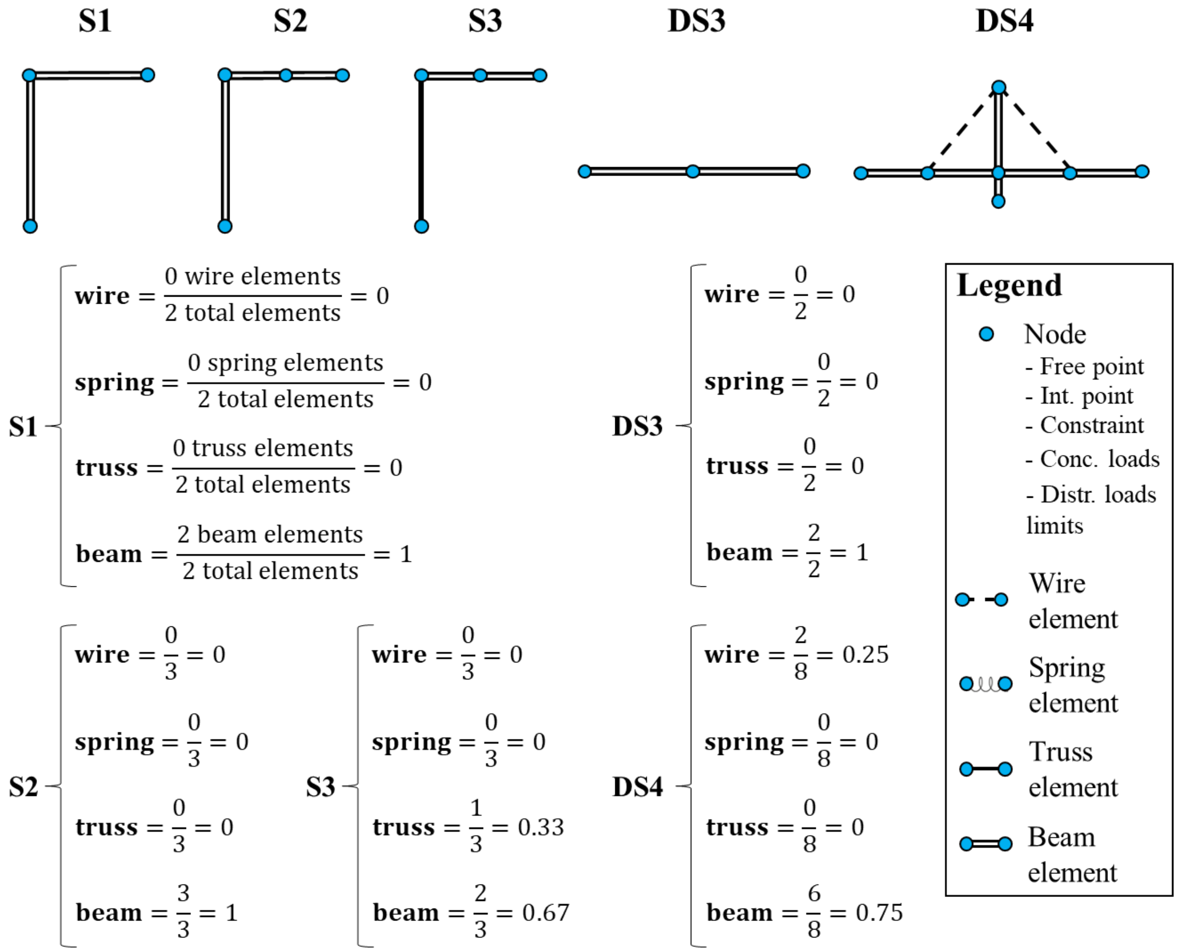
- The feature “straight” is the ratio of the number of constraints linked by adjacent segments to the total number of constraints. For example, in DS4, the two constraints located on the left side and on the right side are linked by the four adjacent horizontal segments, leading to 0.67 (2/3). Instead, in S1, S2 and S3, we have no adjacent segments, which means that the value of this feature is always 0.
- The feature “extremity” is the ratio between the number of constraints coinciding with the contour vertices and the total number of constraints.

We can see that S1, S2 and S3 are now distinguished by the configuration of the boundary conditions, since they are characterised by different values of the CF “extremity”.

Group CFs^3 = {wire, spring, truss, beam}. As shown in Fig. 83, the structures of Fig. 80 are represented as ensembles of elements delimited by nodes, thus enabling to visualize the used behaviour models (cf. Fig. 78). The element numbers indicated in Fig. 83 are assigned arbitrarily. The concept of node is inspired to the finite element method (FEM, cf. [119]), and it includes:

- Free points, intersection points and constraint positions (cf. Fig. 82).
- The application points of the concentrated loads (if any).
- The limits of the distributed load loads (if any).

For example, elements 2 and 3 in S3 are separated by the limit of the distributed load (cf. Fig. 80), while elements 1 and 2 in DS3 are separated by application point of the concentrated load (cf. Fig. 80).



$$\mathbf{S1}\begin{cases}\mathbf{wire}=\dfrac{0\text{ wire elements}}{2\text{ total elements}}=0\\[2ex] \mathbf{spring}=\dfrac{0\text{ spring elements}}{2\text{ total elements}}=0\\[2ex] \mathbf{truss}=\dfrac{0\text{ truss elements}}{2\text{ total elements}}=0\\[2ex] \mathbf{beam}=\dfrac{2\text{ beam elements}}{2\text{ total elements}}=1\end{cases}$$

$$\mathbf{DS3}\begin{cases}\mathbf{wire}=\dfrac{0}{2}=0\\[2ex] \mathbf{spring}=\dfrac{0}{2}=0\\[2ex] \mathbf{truss}=\dfrac{0}{2}=0\\[2ex] \mathbf{beam}=\dfrac{2}{2}=1\end{cases}$$

$$\mathbf{S2}\begin{cases}\mathbf{wire}=\dfrac{0}{3}=0\\[2ex] \mathbf{spring}=\dfrac{0}{3}=0\\[2ex] \mathbf{truss}=\dfrac{0}{3}=0\\[2ex] \mathbf{beam}=\dfrac{3}{3}=1\end{cases}$$

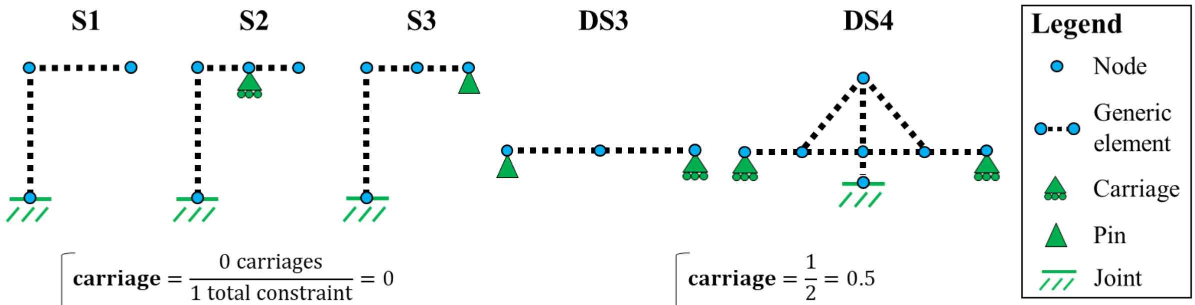
$$\mathbf{S3}\begin{cases}\mathbf{wire}=\dfrac{0}{3}=0\\[2ex] \mathbf{spring}=\dfrac{0}{3}=0\\[2ex] \mathbf{truss}=\dfrac{1}{3}=0.33\\[2ex] \mathbf{beam}=\dfrac{2}{3}=0.67\end{cases}$$

$$\mathbf{DS4}\begin{cases}\mathbf{wire}=\dfrac{2}{8}=0.25\\[2ex] \mathbf{spring}=\dfrac{0}{8}=0\\[2ex] \mathbf{truss}=\dfrac{0}{8}=0\\[2ex] \mathbf{beam}=\dfrac{6}{8}=0.75\end{cases}$$

Fig. 83: Definition criteria of the CFs in the third group (CFs^3 = {wire, spring, truss, beam}, cf. Table 84).

According to Fig. 83, each CF is easily defined as the ratio between the number of the elements corresponding to the type indicated by the same CF and the total number of elements. Consequently, if a type has been assigned to all elements, the sum of the CFs “wire”, “spring”, “truss” and “beam” in each structure is necessarily equal to 1.

Group CFs^4 = {carriage, pin, joint}. As shown in Fig. 84, the structures are considered as ensembles of generic elements delimited by the same nodes defined in Fig. 83, and where it is possible to distinguish the different types of used constraints (cf. Fig. 80). Each CF is easily defined as the ratio between the number of the constraints corresponding to the type indicated by the same CF and the total number of constraints. Consequently, the sum of the CFs “carriage”, “pin” and “joint” in each structure is necessarily equal to 1.

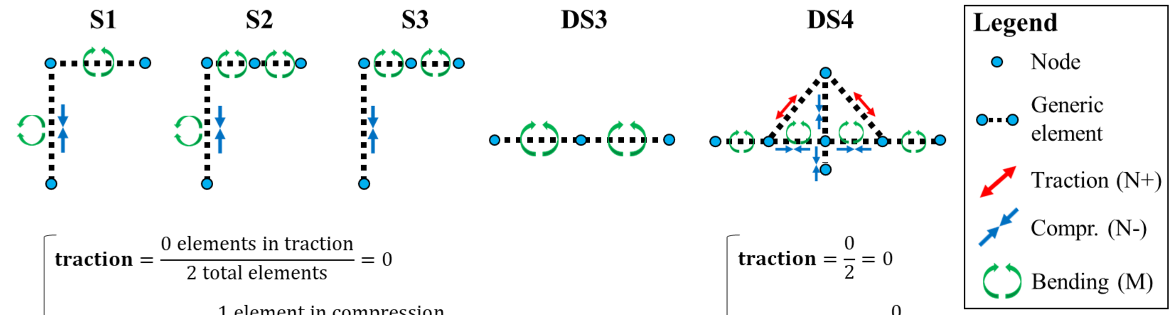


$$\text{S1}\left\{\begin{array}{l}\textbf{carriage} = \dfrac{0 \text{ carriages}}{1 \text{ total constraint}} = 0\\[2ex] \textbf{pin} = \dfrac{0 \text{ pins}}{1 \text{ total constraint}} = 0\\[2ex] \textbf{joint} = \dfrac{1 \text{ joint}}{1 \text{ total constraint}} = 1\end{array}\right. \qquad \text{DS3}\left\{\begin{array}{l}\textbf{carriage} = \dfrac{1}{2} = 0.5\\[2ex] \textbf{pin} = \dfrac{1}{2} = 0.5\\[2ex] \textbf{joint} = \dfrac{0}{2} = 0\end{array}\right.$$

$$\text{S2}\left\{\begin{array}{l}\textbf{carriage} = \dfrac{1}{2} = 0.5\\[2ex] \textbf{pin} = \dfrac{0}{2} = 0\\[2ex] \textbf{joint} = \dfrac{1}{2} = 0.5\end{array}\right. \qquad \text{S3}\left\{\begin{array}{l}\textbf{carriage} = \dfrac{0}{2} = 0\\[2ex] \textbf{pin} = \dfrac{1}{2} = 0.5\\[2ex] \textbf{joint} = \dfrac{1}{2} = 0.5\end{array}\right. \qquad \text{DS4}\left\{\begin{array}{l}\textbf{carriage} = \dfrac{2}{3} = 0.67\\[2ex] \textbf{pin} = \dfrac{0}{3} = 0\\[2ex] \textbf{joint} = \dfrac{1}{3} = 0.33\end{array}\right.$$

Fig. 84: Definition criteria of the CFs in the fourth group (CFs^4 = {carriage, pin, joint}, cf. Table 84).

Group CFs^5 = {traction, compression, bending}. As shown in Fig. 85, the structures are considered as ensembles of generic elements delimited by the same nodes defined in Fig. 83, visualizing the internal solicitations due to the external loads (cf. Fig. 80). Three types of internal solicitation are considered: traction normal effort (N+), compression normal effort (N-) and bending moment (M), whose respective CFs are defined as illustrated in Fig. 85. We can remark that the sum of the CFs “traction”, “compression” and “bending” in each structure is not necessarily equal to 1, since a same element can be solicited by both traction and bending or by both compression and bending.



$$\text{S1}\left\{\begin{array}{l}\textbf{traction} = \dfrac{0 \text{ elements in traction}}{2 \text{ total elements}} = 0\\[2ex] \textbf{compresssion} = \dfrac{1 \text{ element in compression}}{2 \text{ total elements}} = 0.5\\[2ex] \textbf{bending} = \dfrac{2 \text{ elements in bending}}{2 \text{ total elements}} = 1\end{array}\right. \qquad \text{DS3}\left\{\begin{array}{l}\textbf{traction} = \dfrac{0}{2} = 0\\[2ex] \textbf{compression} = \dfrac{0}{2} = 0\\[2ex] \textbf{bending} = \dfrac{2}{2} = 1\end{array}\right.$$

$$\text{S2}\left\{\begin{array}{l}\textbf{traction} = \dfrac{0}{3} = 0\\[2ex] \textbf{compression} = \dfrac{1}{3} = 0.33\\[2ex] \textbf{joint} = \dfrac{3}{3} = 1\end{array}\right. \qquad \text{S3}\left\{\begin{array}{l}\textbf{traction} = \dfrac{0}{2} = 0\\[2ex] \textbf{compression} = \dfrac{1}{3} = 0.33\\[2ex] \textbf{joint} = \dfrac{2}{3} = 0.67\end{array}\right. \qquad \text{DS4}\left\{\begin{array}{l}\textbf{traction} = \dfrac{2}{8} = 0.25\\[2ex] \textbf{compression} = \dfrac{4}{8} = 0.5\\[2ex] \textbf{bending} = \dfrac{4}{8} = 0.5\end{array}\right.$$

Fig. 85: Definition criteria of the CFs in the fifth group (CFs^5 = {traction, compression, bending}, cf. Table 84).

All the presented criteria consist of ratios defined in the range [0;1], and whose description is summarised in Table 88 in terms of numerator and denominator. The ensemble of these criteria enables the user to encode the initial idea into a structural BOW model (cf. Fig. 79). Moreover, the originality of the defined criteria is that the determined CFs’ values do not necessarily identify a unique structure, but an ensemble of structure. This aspect will be exploited in section 3.2.3, during the generation of the calibration set for the ANNs.

Table 88: Definition criteria of the CFs' values: description of numerator and denominator in the ratios (cf. Fig. 81-Fig. 85).

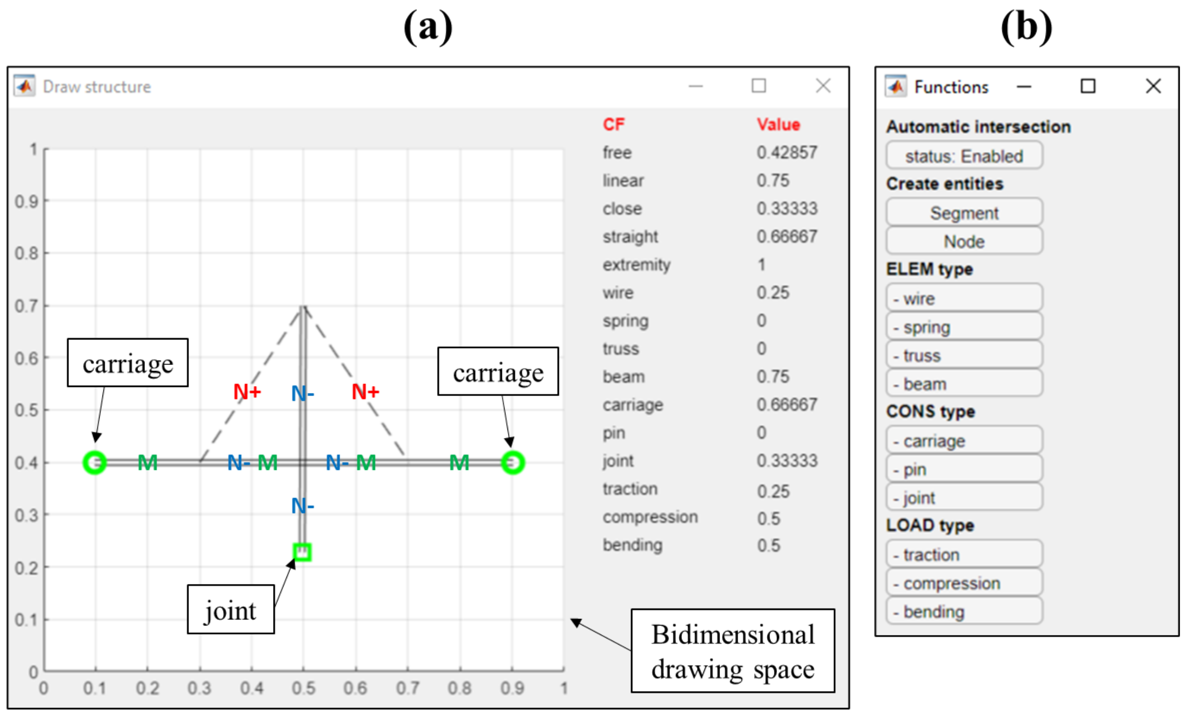
CFs' group	CF	Numerator	Denominator	Reference figure
CFs^1	free	Number of free points	Sum of the numbers of free points and intersection points	Fig. 81
	linear	Total number of adjacent segments	Total number of segments	
	close	Number of structurally closed portions	Total number of portions	
CFs^2	straight	Number of constraints linked by adjacent segments	Total number of constraints	Fig. 82
	extremity	Number of constraints coinciding with the contour vertices		
CFs^3	wire	Number of wire elements	Total number of elements	Fig. 83
	spring	Number of spring elements		
	truss	Number of truss elements		
	beam	Number of beam elements		
CFs^4	carriage	Number of carriages	Total number of constraints	Fig. 84
	pin	Number of pins		
	joint	Number of joints		
CFs^5	traction	Number of elements in traction (N+)	Total number of elements	Fig. 85
	compression	Number of elements in compression (N-)		
	bending	Number of elements in bending (M)		

3.2.2.2. An original drawing tool to promote hand-sketching

The comprehension of the criteria to define the CFs (cf. Table 88) and their application may be a little challenging for a new user of the demonstrator. All the criteria were thus integrated in an original drawing tool inspired to CAD programs and implemented in MATLAB [48]. The idea of this tool is actually twofold:

- Facilitating and speeding up the definition of the CFs' values when conducting the initial encoding step in the NLP algorithm (cf. Fig. 79). A base knowledge of the criteria in Table 88 is clearly required for using the drawing tool.
- In accordance with our purposes (cf. Table 9), developing a tool of "aided hand-sketching" free from the rules of mechanics and which leaves more space to improvisation, for helping the user overcoming his cognitive bias (cf. Table 8).

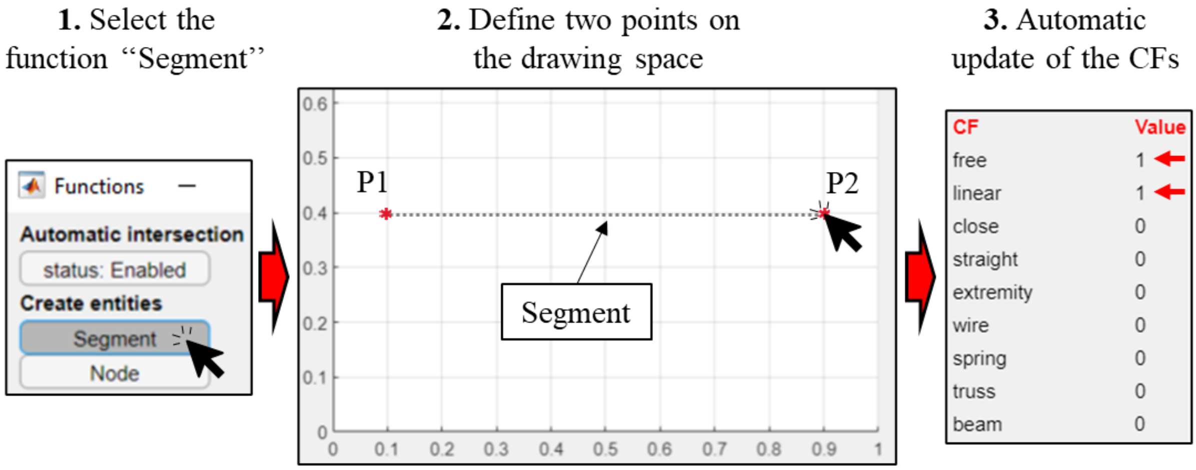
The program interface is composed by the two windows shown in Fig. 86. The window in Fig. 86a provides a bidimensional space where the user can draw the structure and the values of the CFs associated to the drawn structure which are automatically calculated by the program. The structure drawn in Fig. 86a corresponds to DS4, and we can see that all CFs are equal to those found in section 3.2.2.1 (cf. Fig. 81-Fig. 85). According to the criteria defined in section 3.2.2.1 (cf. Table 88), the window in Fig. 86b is used to select the functions for creating segments and nodes (cf. Fig. 81 and Fig. 83, respectively), for assigning the element types, for creating the constraints and for defining the internal loads. All functions are provided as buttons (cf. Fig. 86b).



CF	Value
free	0.42857
linear	0.75
close	0.33333
straight	0.66667
extremity	1
wire	0.25
spring	0
truss	0
beam	0.75
carriage	0.66667
pin	0
joint	0.33333
traction	0.25
compression	0.5
bending	0.5

Fig. 86: Interface windows of the original drawing tool: (a) drawing space and CFs' values; (b) functions [48].

The use of the implemented drawing tool is very intuitive, and it is briefly illustrated as follows. For example, let use draw the horizontal part of DS4 (cf. Fig. 86a), which is characterised by four adjacent beam elements (cf. Fig. 83). We must first draw the horizontal segment representing the whole horizontal part of DS4, following the procedure in Fig. 87. The function "Segment" is selected by the user, who thus defines two points on the drawing space. The segment is created by the program after defining the second point, and the CFs are automatically updated. In a similar way, the procedures to assign element types and to create constraints are shown in Fig. 88. The procedure to assign the internal load type is analogous to the procedure a (cf. Fig. 88). It is important to remark that the order whom operations a and b (cf. Fig. 88) are executed is irrelevant.



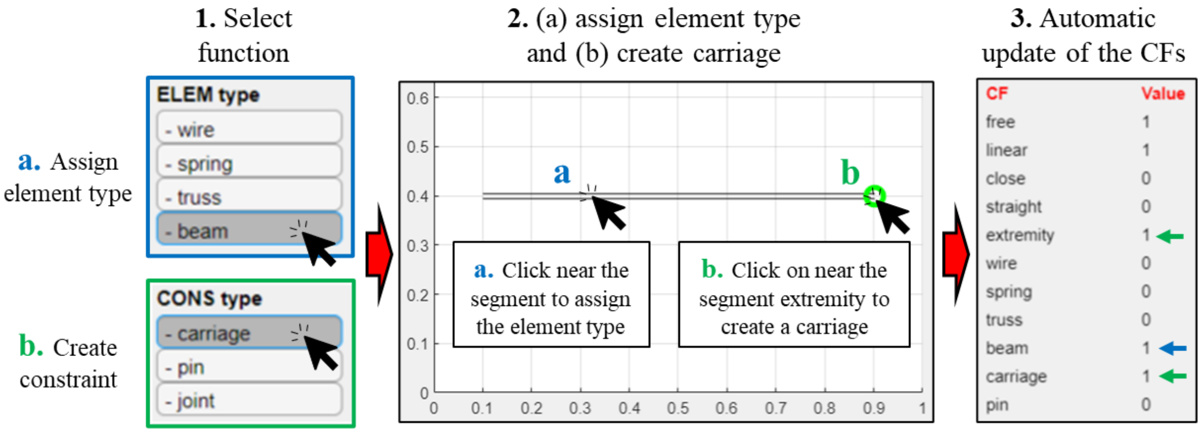
1. Select the function "Segment"

2. Define two points on the drawing space

3. Automatic update of the CFs

CF	Value
free	1
linear	1
close	0
straight	0
extremity	0
wire	0
spring	0
truss	0
beam	0

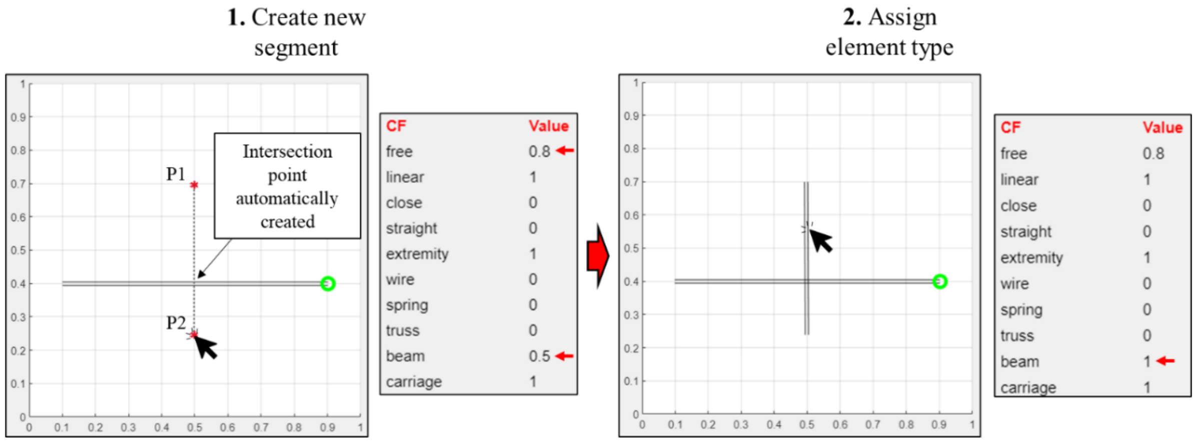
Fig. 87: Creating the horizontal part of DS4 with the drawing tool (cf. Fig. 86).


1. Select function
2. (a) assign element type and (b) create carriage
3. Automatic update of the CFs
a. Assign element type
b. Create constraint
ELEM type
- wire
- spring
- truss
- beam
CONS type
- carriage
- pin
- joint
a
b
a. Click near the segment to assign the element type
b. Click on near the segment extremity to create a carriage
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

CF	Value
free	1
linear	1
close	0
straight	0
extremity	1 ←
wire	0
spring	0
truss	0
beam	1 ←
carriage	1 ←
pin	0

Fig. 88: Procedures (a) to define assign element type and (b) to create a constraint (cf. Fig. 86).

We can now proceed to create the vertical part of DS4 (cf. Fig. 86a) as illustrated in Fig. 89. Based on Fig. 87, the new vertical segment P1P2 is first created by defining two points on the drawing space. The function of automatic intersection (cf. Fig. 86b) guarantees the automatic creation of the intersection point between the horizontal part and the vertical part when the new segment is drawn. Indeed, remark that the feature "free" is updated to 0.8 (step 1, cf. Fig. 89), since the new structure involves 1 intersection point and 4 free points (cf. Fig. 81). Moreover, the feature "beam" is updated to 0.5 (step 1, cf. Fig. 89), since the new structure involves 2 beam elements (horizontal) and 2 generic elements (vertical), i.e. whose type has not been assigned yet. The function of automatic intersection can be easily disabled by pushing on the corresponding button (cf. Fig. 86b) if the user wants to avoid creating an intersection point.



CF	Value
free	0.8 ←
linear	1
close	0
straight	0
extremity	1
wire	0
spring	0
truss	0
beam	0.5 ←
carriage	1

CF	Value
free	0.8
linear	1
close	0
straight	0
extremity	1
wire	0
spring	0
truss	0
beam	1 ←
carriage	1

Fig. 89: Steps to create the vertical part of DS4 (cf. Fig. 86).

After creating the new vertical segment, the element type can be assigned as shown in Fig. 88. Remark that the feature "beam" is updated to 1 (step 2, cf. Fig. 89), since the whole structure is now characterised by beam elements. We have seen that the drawing methodology proposed in the implemented tool is intuitive and is free from any mechanical rule regarding the statics of the considered structure. For instance, we may draw structures without necessarily assigning an element type to all segments. We believe that this could favour creative improvisation, which is a fundamental component of the BDPs (cf. Table 5), enabling the user to transcend technical details. This property of the implemented tool is object of the future developments of the demonstrator (which are discussed in section 3.2.5), and its benefits to conceptual design thus still need to be better studied and evaluated.

3.2.2.3. Definition and interpretation of the SPs

Based on Fig. 79, the SPs are sentences provided in an original abbreviated form, to facilitate the construction of the LTR models. As schematized in Fig. 90, the SPs are sentences composed by two parts. The first part involves an action suggested to satisfy a required specification, while the second part gives further indications for applying the action to the initial idea, as for example the type of element which must be affected by the action. As seen Fig. 79, the second part can be provided or not, depending on the complexity of the expressed conceptual solution. If the second part is not provided, the suggested action can be applied to the whole initial idea at the user's discretion.

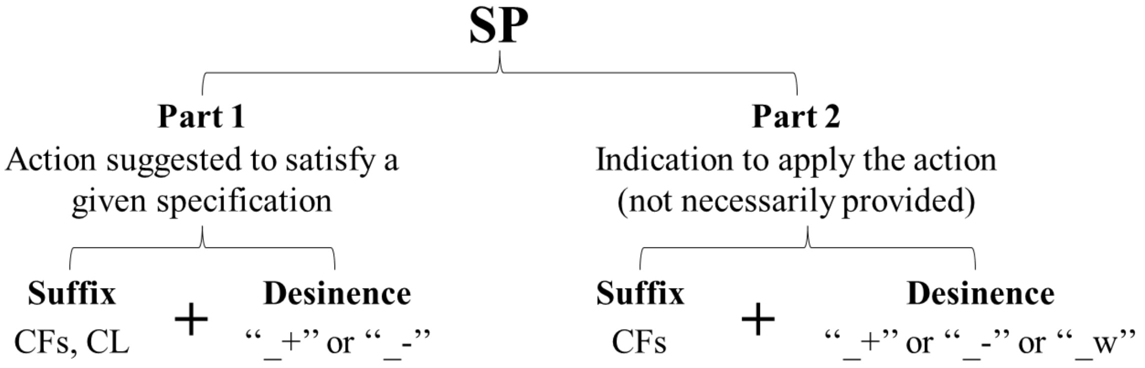

SP
Part 1
Action suggested to satisfy a given specification
Part 2
Indication to apply the action (not necessarily provided)
Suffix
CFs, CL
+
Desinence
"_+" or "_-"
Suffix
CFs
+
Desinence
"_+" or "_-" or "_w"

Fig. 90: Scheme of a SP.

According to Fig. 90, both parts of a SP are composed by a suffix and a desinence. It is assumed that the suffix of the first part can be formed by both CFs (cf. Table 84) and words of the CL (cf. Table 86), while the suffix of the second part by only CFs. Three different desinences are assumed:

- "_+" (plus) indicates that the object of the corresponding suffix should be increased (or added);
- "_-" (minus) indicates that the object of the corresponding suffix should be decreased (or removed);
- "_w" (where) is used to designate the corresponding suffix as a location.

Different types of conceptual solutions can be obtained, depending on how the desinences are employed in the two parts of the SP. The six assumed types of SPs are reported in Table 89, where the suffixes of action (part 1) and indication (part 2) are respectively indicated with "object1" and "object2".

Table 89: Assumed types of SP and respective interpretation.

Type	Action (part 1)	Indication (part 2)	Interpretation
1. General increase (addition)	object1_+		Increase (add) object1
2. General decrease (removal)	object1_-		Decrease (remove) object1
3. Local increase (addition)	object1_+	object2_w	Increase (add) object1 in/at/of object2
4. Local decrease (removal)	object1_-	object2_w	Decrease (remove) object1 in/at/of object2
5. Replacement	object1_+	object2_-	Add object1 in place of object2
6. Structural addition	object1_+	object2_+	Add object1 fixed to object2 (object2 is a new constraint to add)

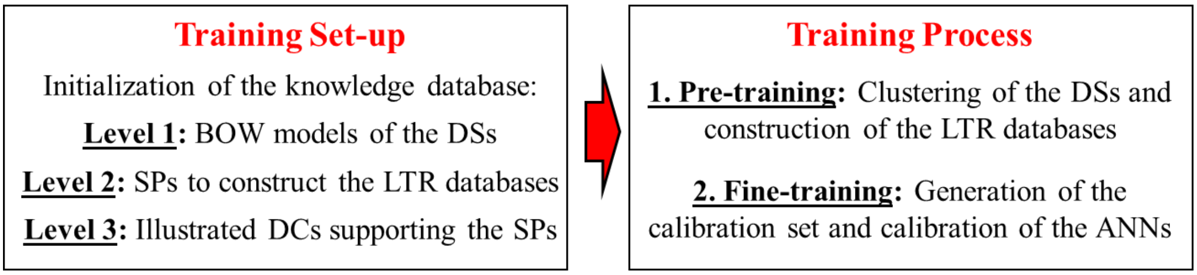
According to Table 89, the SPs provided in Fig. 79, i.e. "A_-" and "hollow_+ bending_w", respectively belong to the types 2 and 3. Notably, the SP "A_-" recommends decreasing the section area to reduce the structural mass. Lacking further indications (part 2), the section area can be reduced both in the horizontal and in the vertical part of the initial idea (cf. Fig. 79). The SP "hollow_+ bending_w" (type 3, cf. Table 89) recommends adding the characteristic of hollow section (cf. Table 86) in the parts of the structure which are solicited to bending. According to Table 89, the SPs belonging to the types 1 and 4 are interpreted in a similar way. Two examples of the types 5 and 6 as follows:

- Type 5: “pin_+ carriage_-”, recommending replacing a carriage with a pin.
- Type 6: “truss_compression_+ pin_+”, suggesting reinforcing the structure by adding a truss in compression fixed to a new pin.

When multiple words are used in the suffix, they are organized in the order of Table 84 and Table 86, in accordance with Fig. 90. In general, the types 1, 2, 3 and 4 can be easily understood and applied to the initial idea thanks to the interpretation provided in Table 89, while the types 5 and 6 involve more complex and specific actions and thus need more exhaustive explanations to be applied. According to the adopted approach (PAAs, cf. Table 18), the types 5 and 6 and some other SPs will be thus supported by illustrated design examples inspired to the effects used in TIPS (cf. Table 16) and based on the DSs (cf. Fig. 78). The illustrated design examples are indicated as supporting DCs (supporting design cases) and will be presented in section 3.2.3.

3.2.3. Training of the NLP algorithm employed in the demonstrator

The training of the NLP algorithm employed in the demonstrator involves the two phases reported in Fig. 91. In the training set-up phase, we initialize the knowledge database of the NLP algorithm. As shown Fig. 91, the knowledge database is composed by three different levels: the BOW models of the DSs, the SPs to construct the LTR databases (cf. Fig. 79) and the DCs supporting some SPs (cf. section 3.2.2.3). The following phase corresponds to the training process of the NLP algorithm (cf. Fig. 62), and it is conducted in a similar way to that of the improved NLP methodology presented in section 3.1.3.1.



Training Set-up

Initialization of the knowledge database:

Level 1: BOW models of the DSs

Level 2: SPs to construct the LTR databases

Level 3: Illustrated DCs supporting the SPs

Training Process

1. Pre-training: Clustering of the DSs and construction of the LTR databases

2. Fine-training: Generation of the calibration set and calibration of the ANNs

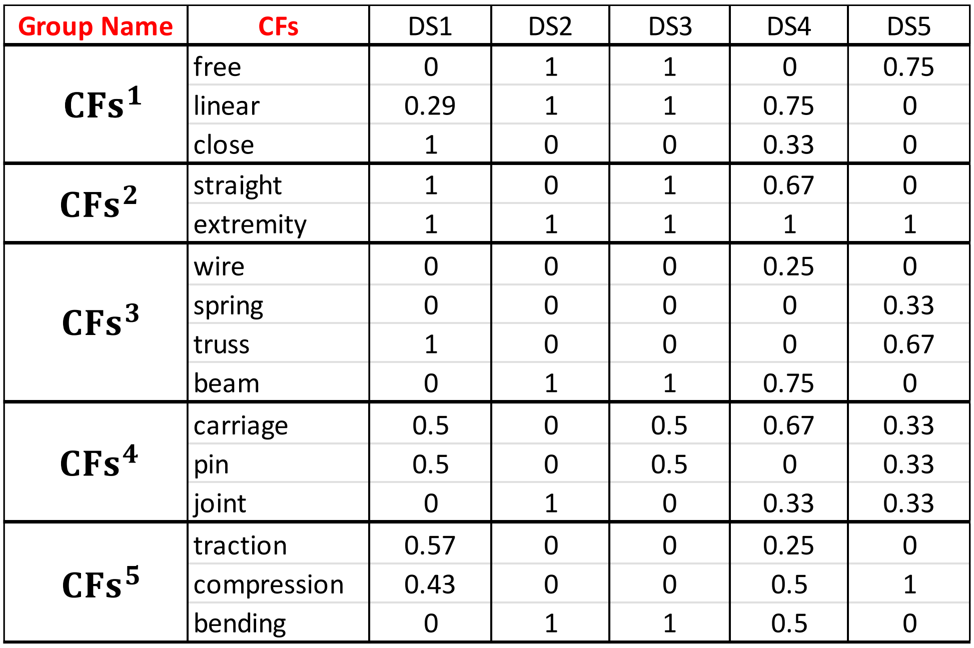
Fig. 91: Steps to train the NLP algorithm (cf. Fig. 62).

This section is organized as follows. The set-up of the knowledge database is described in section 3.2.3.1. The pre-training step is thus presented in section 3.2.3.2, while the fine-training step in section 3.2.3.3.

3.2.3.1. Training set-up: initialization of the knowledge database

The overall knowledge database is initialized into a unique Excel file including three different datasheets, one for each level. The three initialized knowledge levels are reported and described one by one as follows.

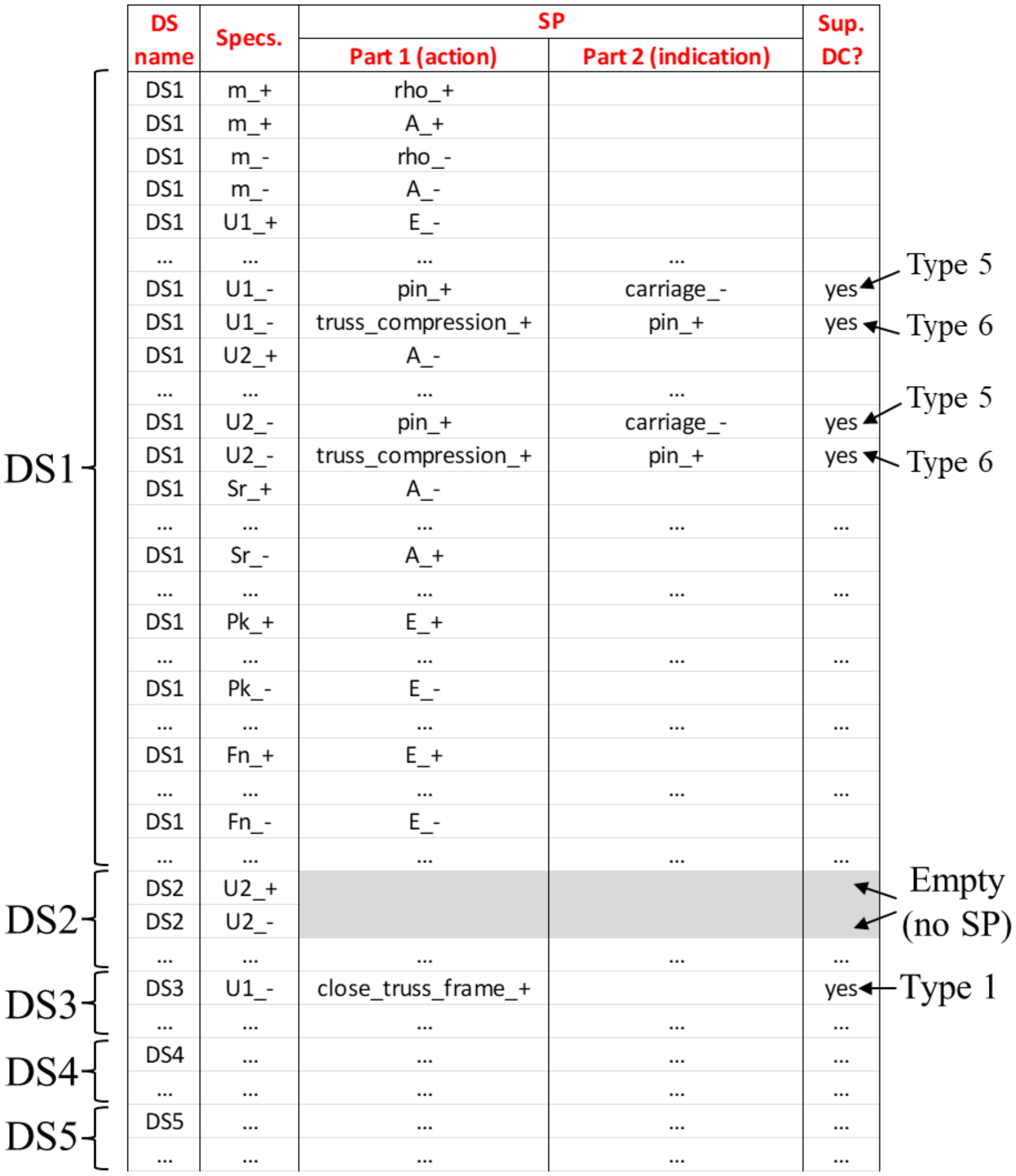
Level 1. The database of the DSs is shown in Fig. 92. The rows contain the CFs in the same order as Table 84, while the columns contain the names of the five considered DSs (cf. Fig. 78). The initialized values of the CFs were calculated with the drawing tool presented in section 3.2.2.2.



Group Name	CFs	DS1	DS2	DS3	DS4	DS5
CFs^1	free	0	1	1	0	0.75
	linear	0.29	1	1	0.75	0
	close	1	0	0	0.33	0
CFs^2	straight	1	0	1	0.67	0
	extremity	1	1	1	1	1
CFs^3	wire	0	0	0	0.25	0
	spring	0	0	0	0	0.33
	truss	1	0	0	0	0.67
	beam	0	1	1	0.75	0
CFs^4	carriage	0.5	0	0.5	0.67	0.33
	pin	0.5	0	0.5	0	0.33
	joint	0	1	0	0.33	0.33
CFs^5	traction	0.57	0	0	0.25	0
	compression	0.43	0	0	0.5	1
	bending	0	1	1	0.5	0

Fig. 92: Excel datasheet of level 1: BOW models of the DSs (cf. Fig. 91).

Level 2. The complete database of level 2 (cf. Fig. 91) includes about 40 different SPs. A section of the initialized SPs' database is shown in Fig. 93, where the first column contains the names of the five DSs initialized in Fig. 92. Based on Table 85 and Fig. 79, 12 different design specifications were defined by combining the 6 EPs assumed and the two desinences "_+" and "_-". For each DS, we generated as many different SPs as possible for satisfying all 12 specifications, based on the indications given in Fig. 90 and Table 89. We can see that the same SP can be used to satisfy different specifications. For example, the SP "truss_compression_+ pin_+" is used in DS1 for decreasing both U1 and U2. The efficacy of all SPs was tested with the finite element method (FEM). The FEM models of the DSs used for this purpose were implemented in ABAQUS [120], and their characteristics are summarised in annex **Errore. L'origine riferimento non è stata trovata.**. Some examples illustrating the verification of the SPs via the FEM models are reported in annex B.2. The empty cells in Fig. 93 indicate that no SP was defined for those specifications. In DS2 (cf. Fig. 78), for example, U2 is equal to zero due to the assumptions in Table 83, and the specifications "U2_+" and "U2_-" cannot be thus satisfied. Finally, the right column of the SP database indicates if the corresponding SPs are supported by one or more DCs (cf. section 3.2.2.3). The supporting DCs are employed for all SPs belonging to the types 5 and 6 (cf. Table 89) and, where necessary, for some SPs belonging to the other types (cf. Fig. 93).


DS1
DS2
DS3
DS4
DS5
Type 5
Type 6
Type 5
Type 6
Empty
(no SP)
Type 1

DS name	Specs.	SP		Sup. DC?
		Part 1 (action)	Part 2 (indication)	
DS1	m_+	rho_+		
DS1	m_+	A_+		
DS1	m_-	rho_-		
DS1	m_-	A_-		
DS1	U1_+	E_-		
...	...	...	...	
DS1	U1_-	pin_+	carriage_-	yes
DS1	U1_-	truss_compression_+	pin_+	yes
DS1	U2_+	A_-		
...	...	...	...	
DS1	U2_-	pin_+	carriage_-	yes
DS1	U2_-	truss_compression_+	pin_+	yes
DS1	Sr_+	A_-		
...	...	...	...	...
DS1	Sr_-	A_+		
...	...	...	...	...
DS1	Pk_+	E_+		
...	...	...	...	...
DS1	Pk_-	E_-		
...	...	...	...	...
DS1	Fn_+	E_+		
...	...	...	...	...
DS1	Fn_-	E_-		
...	...	...	...	...
DS2	U2_+			
DS2	U2_-			
...	...	...	...	...
DS3	U1_-	close_truss_frame_+		yes
...	...	...	...	...
DS4	...	...	...	...
...	...	...	...	...
DS5	...	...	...	...
...	...	...	...	...

Fig. 93: Section of the Excel datasheet of level 2: list of the SPs to construct the LTR databases (cf. Fig. 79 and Fig. 91).

Level 3. The function of the DCs is to illustrate possible ways of applying the SPs to the DSs, supporting the interpretation of the SPs (cf. Table 89). The DCs are based on the DSs, and they are verified with the same FEM models used for the SPs (cf. annex B.2, examples 3 and 4). The generated DCs were organized in a morphological matrix inspired to that used in TIPS for sub-functions and effects (cf. Table 16). A section of the initialized morphological matrix is shown in Fig. 94. The rows of the matrix contain the five DSs (cf. Fig. 78), while the columns contain the SPs for which the presence of supporting DCs was indicated in the initialized SPs' database (cf. Fig. 93). For example, the DC corresponding to DS1 and to the SP "truss_compression_+ pin_+" (type 6, cf. Table 89) shows the evolution of DS1 by adding two new trusses in compression fixed to two new pins. To define this DC, we assumed that the new added pins are aligned to the constraints already existing. Notably, same DCs can support different SPs, as well same SPs can be used to satisfy different specifications (cf. Fig. 93). Moreover, more than one DC can be contained in the same cell of the morphological matrix, since there could be more than one possibility of applying the same SP to the same DS. The grey colour indicates the empty cells.

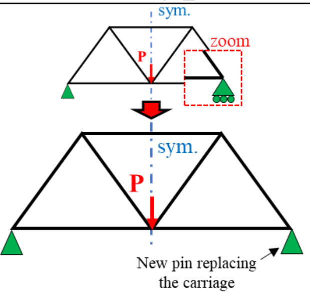
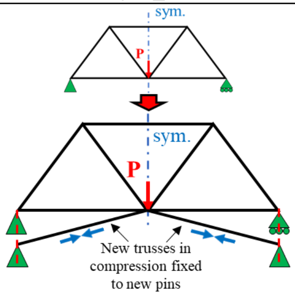
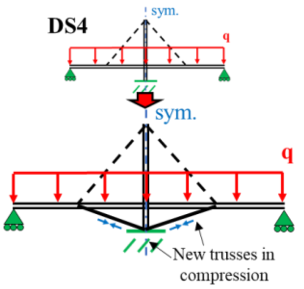
	pin_+	truss_compression_+	…	truss_compression_+	…
	carriage_-	pin_+	…		…
DS1	 sym. zoom P sym. P New pin replacing the carriage	 sym. P sym. P New trusses in compression fixed to new pins	…		…
DS2	…	…	…	…	…
DS3	…	…	…	…	…
DS4	…	…	…		…
DS5	…	…	…	…	…

Fig. 94: Section of the Excel datasheet of level 3: morphological matrix containing the DCs (cf. Fig. 91).

The knowledge database referred to the single DS1 is finally schematized in Fig. 95, which is based on Fig. 18. A similar schematization can be clearly adopted for all the other DSs. The schematization in Fig. 95 enables to better visualize the three knowledge levels presented in this section.

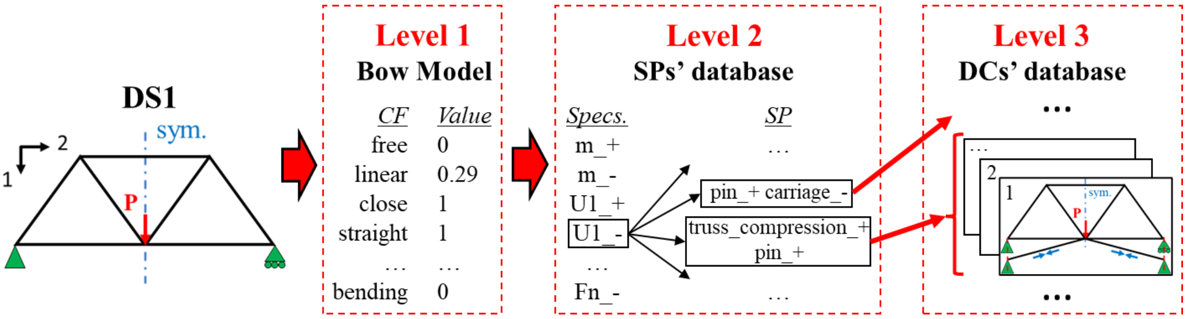


Fig. 95: Schematization of the knowledge database referred to the single DS1 (cf. Fig. 91).

According to the adopted approach (cf. Table 18), the knowledge database of the NLP algorithm (cf. Fig. 92-Fig. 94) involves an original use of the natural language (CFs and SPs) supported by illustrated DCs. Notably, the efficacy of the initialized SPs was tested by means of FEM models, but these only represent a verification tool exterior to the implemented NLP methodology, and they are not used in the demonstrator to provide responses.

3.2.3.2. Pre-training: clustering and construction of the LTR databases

The pre-training of the current NLP algorithm is based on that presented in section 3.1.3.1, considering the DSs as training samples. The datasets to cluster are generated from the database of level 1 (cf. Fig. 92), and they are schematized in Table 90, based on (cf. Table 78).

Table 90: Schematization of the training datasets to cluster corresponding to the five adopted groups of CFs (cf. Table 78 and Fig. 92).

	Dataset 1 (CFs^1)			**Dataset 2 (CFs^2)**		**Dataset 3 (CFs^3)**			**…**	**Dataset 5 (CFs^5)**		
S. #	**free**	**…**	**close**	**straight**	**extr.**	**wire**	**…**	**beam**	**…**	**traction**	**…**	**bending**
DS1	0	…	1	1	1	0	…	0	…	0.57	…	0
DS2	1	…	0	0	1	0	…	1	…	0	…	1
DS3	1	…	0	1	1	0	…	1	…	0	…	1
DS4	0	…	0.33	0.67	1	0.25	…	0.75	…	0.25	…	0.5
DS5	0.75	…	0	0	1	0	…	0	…	0	…	0

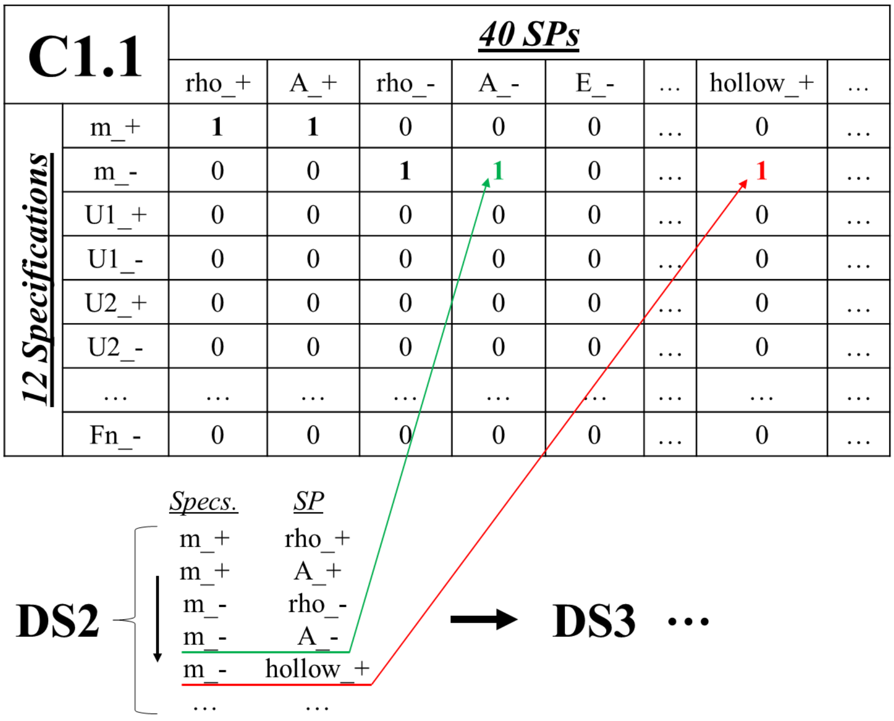
Based on section 3.1.3.1, five different clustering procedures (one for each training dataset, cf. Table 90) were conducted with the implemented GA. In each clustering procedure, the number of clusters is increased from K=2 to K=4[24]. The set-up parameters of the GA correspond to those reported in annex A.2 (from K=2 to K=4). For brevity the obtained curves K-R_D (cf. Fig. 69) are summarised in annex B.3. The five resulting partitioning levels, the corresponding clusters and the respective ASs are reported in Table 91. For each partitioning, the number of samples in the clusters is uniform, and there is thus no need to apply the redistribution process of Fig. 70. The pertinence of the obtained results can be easily assured by comparing the CFs' values of the DSs in each cluster to the corresponding ASs. In $P_1^{K=3}$, for example, C1.1 contains DS2 and DS3, both characterised by unitary values of the CFs "free" and "linear", and by null values of the CF "close". We can see that the corresponding AS1.1 is characterised by the same features (cf. Table 91). Similar considerations can be made for each cluster in each partitioning.

[24] For K=5 the ASs to determine directly correspond to the five DSs

Table 91: Resulting partitioning levels, corresponding clusters and determined ASs (cf. annex B.3).

CFs group	Partitioning	Cluster	DS#	AS	ASs' features (CFs)			
CFs^1	$P_1^{K=3}$				free	linear	close	
		C1.1	2, 3	AS1.1	1	1	0	
		C1.2	5	AS1.2	0.73	0	0	
		C1.3	1, 4	AS1.3	0	0.53	0.67	
CFs^2	$P_2^{K=2}$				straight	extremity		
		C2.1	1, 3, 4	AS2.1	0.87	1		
		C2.2	2, 5	AS2.2	0	1		
CFs^3	$P_3^{K=2}$				wire	spring	truss	beam
		C3.1	1, 5	AS3.1	0	0.13	0.87	0
		C3.2	2, 3, 4	AS3.2	0.07	0	0	0.93
CFs^4	$P_4^{K=3}$				carriage	pin	joint	
		C4.1	1, 3	AS4.1	0.53	0.53	0	
		C4.2	2	AS4.2	0	0	1	
		C4.3	4, 5	AS4.3	0.47	0.13	0.33	
CFs^5	$P_5^{K=3}$				traction	comp.	bending	
		C5.1	2, 3	AS5.1	0	0	1	
		C5.2	5	AS5.2	0	1	0	
		C5.3	1, 4	AS5.3	0.4	0.47	0.27	

The LTR models involved in the NLP algorithm were shown in Fig. 79. The construction process of the LTR models is based on Fig. 65, and it is illustrated in Fig. 96 for the cluster C1.1 (cf. Table 91). The generic LTR model is first initialized as a matrix of zeros, whose rows and columns respectively contain the 12 specifications and the 40 different SPs available in the whole database of level 2 (cf. Fig. 93). This assures that all the generated LTR models share the same number of rows and the same number of columns. As shown in Fig. 96, the value 1 is assigned to every specification-SP couple corresponding to the DSs contained in the considered cluster. For repeated specification-SP couples, the value 1 in the target cell is maintained. The constructed LTR models are finally organized into a three-dimensional LTR database (cf. Fig. 79). Five different LTR databases are obtained, i.e. one for each partitioning (cf. Fig. 79 and Table 91).



C1.1	40 SPs							
12 Specifications	rho_+	A_+	rho_-	A_-	E_-	…	hollow_+	…
m_+	**1**	**1**	0	0	0	…	0	…
m_-	0	0	**1**	**1**	0	…	**1**	…
U1_+	0	0	0	0	0	…	0	…
U1_-	0	0	0	0	0	…	0	…
U2_+	0	0	0	0	0	…	0	…
U2_-	0	0	0	0	0	…	0	…
…	…	…	…	…	…	…	…	…
Fn_-	0	0	0	0	0	…	0	…

Specs.	SP
m_+	rho_+
m_+	A_+
m_-	rho_-
m_-	A_-
m_-	hollow_+
…	…

Fig. 96: Construction process of the LTR model for the cluster C1.1 = {DS2, DS3} (cf. Table 91).

3.2.3.3. Fine-training: generation of the calibration set and calibration of the ANNs

Unlike section 3.1.3.1, we cannot generate the current calibration set with the process in Fig. 67a, due to the very low number of training samples (i.e. the five DSs). If we did, we would generate a lot of repeated calibration samples, involving a concrete risk of biasing the calibration process (cf. Table 35). Based on Fig. 67a, an alternative process for generating the calibration set was thus developed for the current demonstrator, and it is illustrated in Fig. 97. This new process is divided into three phases. The first one consists of generating a fictitious design sample (FDS), characterised by random CFs' values limited in the range [0;1]. According to the defined CFs' criteria (cf. Table 88), the generated FDS represents an ensemble of hypothetic DSs sharing the same CFs' values. In the second phase, the five target contexts (one for each partitioning) of the new FDS are determined by calculating the Euclidean distance (cf. Eq. (13)) between the same FDS and the ASs (cf. Table 91). For each partitioning, the distance is calculated as function of the CFs in the corresponding CFs' group. For $P_1^{K=3}$, for example, the three distances between FDS and the artificial samples AS1.1, AS1.2 and AS1.3 are evaluated as function of the the CFs "free","linear" and "close", and the minimum distance identifies the target context (cf. Fig. 97). In the third step, the FDS is submitted to masking (cf. Table 71). Based on Fig. 67a, each CF has the 15% probability to be masked, and when this happens the corresponding value is set to zero.

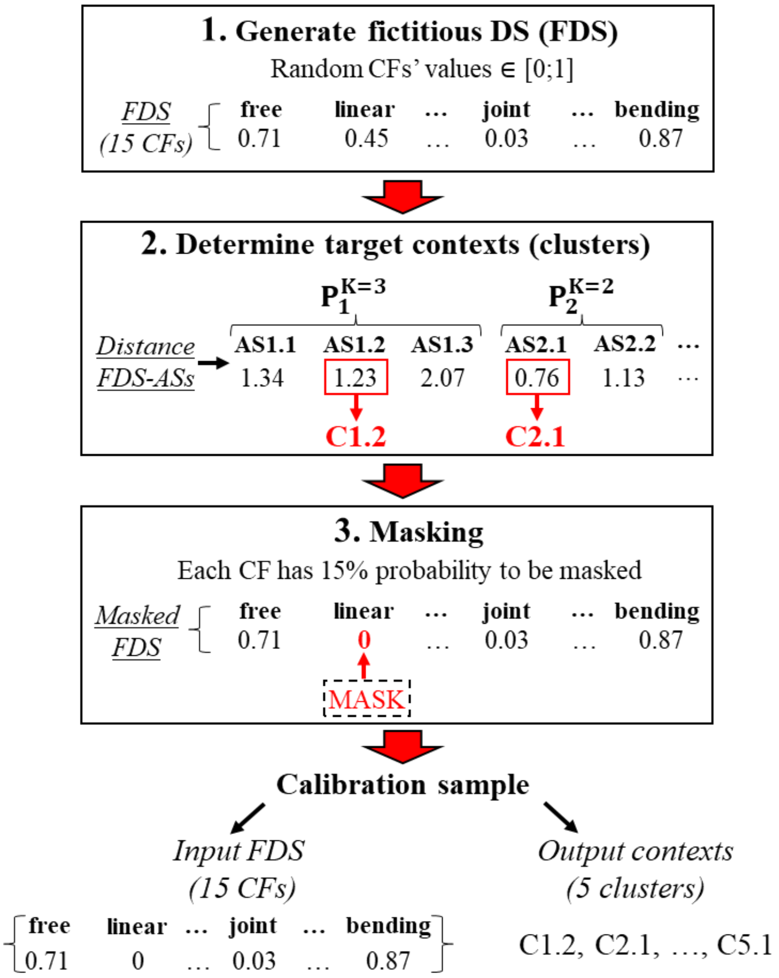

1. Generate fictitious DS (FDS)
Random CFs' values ∈ [0;1]
FDS (15 CFs)
free linear … joint … bending
0.71 0.45 … 0.03 … 0.87
2. Determine target contexts (clusters)
$P_1^{K=3}$
$P_2^{K=2}$
Distance FDS-ASs
AS1.1 AS1.2 AS1.3 AS2.1 AS2.2 …
1.34 1.23 2.07 0.76 1.13 …
C1.2
C2.1
3. Masking
Each CF has 15% probability to be masked
Masked FDS
free linear … joint … bending
0.71 0 … 0.03 … 0.87
MASK
Calibration sample
Input FDS (15 CFs)
Output contexts (5 clusters)
free linear … joint … bending
0.71 0 … 0.03 … 0.87
C1.2, C2.1, …, C5.1

Fig. 97: Generation process of the calibration samples in the current demonstrator (cf. Fig. 67a).

As shown in Fig. 97, the finally obtained calibration sample is thus composed by an input part, i.e. the FDS, and by an output part, i.e. the five target contexts. As in section 3.1.3.1, the generated calibration set consists of 2000 different samples, and it is reported in Table 92. In this case, we have five different sets of calibration outputs, i.e. one for each partitioning (cf. Table 91). Five different ANNs must be thus calibrated, all characterised by 15 input neurons (as the 15 CFs). For each ANN, the number of output neurons is equal to the K of the corresponding partitioning (cf. Table 92), while the number of hidden neurons is determined during the calibration process.

Table 92: Calibration set (cf. Table 81).

	Calibration input				**Cal. output 1 ($P_1^{K=3}$)**				…	**Cal. output 5 ($P_5^{K=3}$)**			
S. #	*FDS (15 CFs)*				*Cont.*	*Probability*			…	*Cont.*	*Probability*		
	free	**linear**	**…**	**bending**		**t_1**	**t_2**	**t_3**	…		**t_1**	**t_2**	**t_3**
1	0.71	0	…	0.87	C1.2	0	1	0	…	C5.1	1	0	0
2	0.06	0.48	…	0	C1.3	0	0	1	…	C5.3	0	0	1
3			…		C1.3	0	0	1	…	C5.2	0	1	0
…	…	…	…	…	…	…	…	…	…	…	…	…	
2000	0	0.69	…	0.23	C1.1	1	0	0	…	C5.3	0	0	1

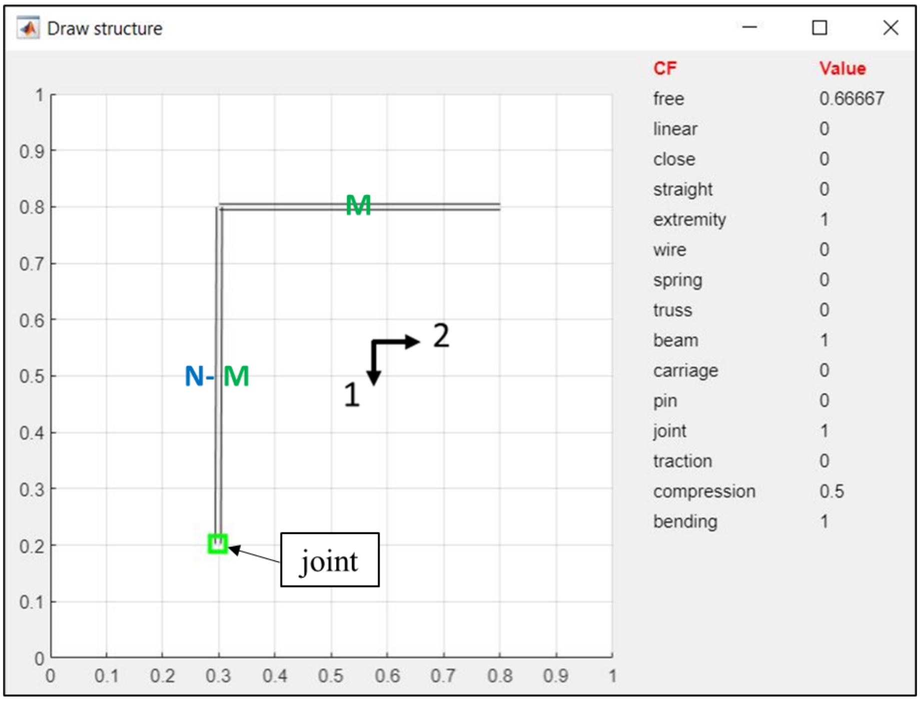
As in section 3.1.3.1, different calibration tests with increasing numbers of hidden neurons were conducted with the implemented ANN for each set of calibration outputs, using the set-up parameters reported in annex A.3. For brevity, the results of the calibration tests in terms of E_{ep}^{v} and the choice of the calibrated ANNs are summarised in annex B.4. The characteristics of the five calibrated ANNs adopted in the NLP algorithm of the demonstrator (cf. Fig. 79) are reported in Table 93, in terms of number of hidden neurons, E_{ep}^{v} and average accuracy. The values of E_{ep}^{v} achieved by the calibrated ANNs are quite similar, and they indicate an average classification accuracy of 91%.

Table 93: Characteristics of the calibrated ANNs adopted for the NLP algorithm of the demonstrator (cf. annex B.4).

Calibrated ANN	**Corresponding partitioning**	**Number of hidden neurons**	**E_{ep}^{v}**	**Average accuracy**
ANN1	$P_1^{K=3}$	10	0.09	≈ 91%
ANN2	$P_2^{K=2}$	5	0.07	
ANN3	$P_3^{K=2}$	15	0.08	
ANN4	$P_4^{K=3}$	10	0.10	
ANN5	$P_5^{K=3}$	15	0.09	

3.2.4. Functioning of the demonstrator and results

The functioning of the demonstrator is illustrated in this section with an example. The initial idea shown in Fig. 79 is assumed as input structure, and it is unknown to the trained NLP algorithm (it is not included in the set of considered DSs, cf. Fig. 78). As shown in Fig. 98, the CFs' values are quickly calculated with the implemented drawing tool (cf. Fig. 86). The assumed input specifications are reported in Table 94, and they involve the reduction of the structural mass and the increasing of the structural stiffness along both directions.



CF	Value
free	0.66667
linear	0
close	0
straight	0
extremity	1
wire	0
spring	0
truss	0
beam	1
carriage	0
pin	0
joint	1
traction	0
compression	0.5
bending	1

Fig. 98: Automatic calculation of the CFs belonging to the input initial idea with the implemented drawing tool (cf. Fig. 86a).

Table 94: Input specifications.

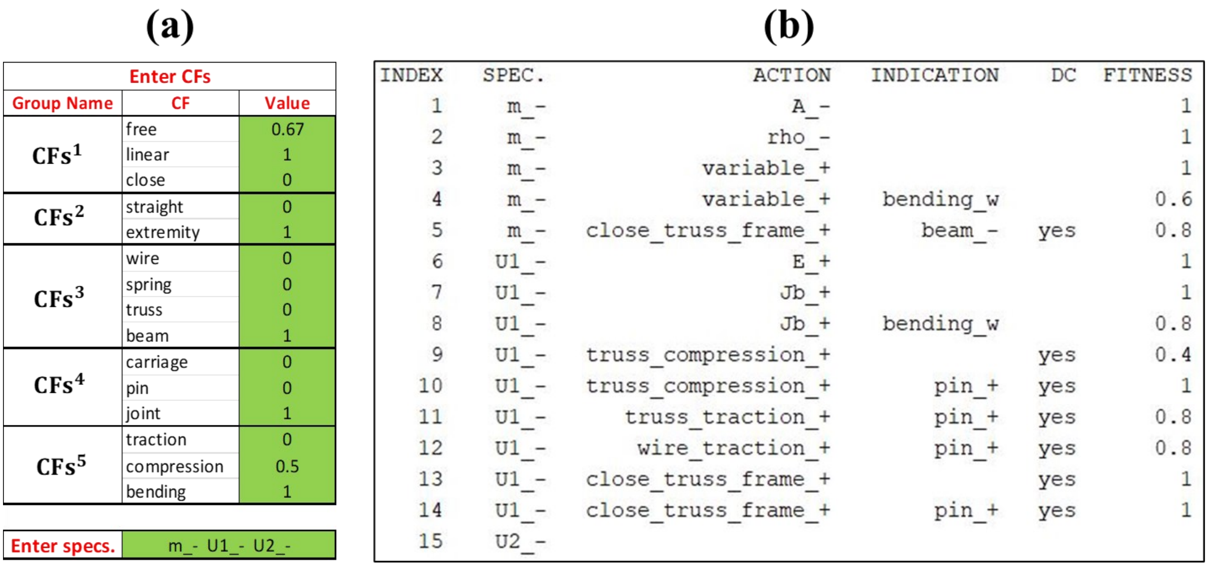
Specs.	Description
m_-	Reduce structural mass
U1_-	Reduce the maximum displacement along direction 1
U2_-	Reduce the maximum displacement along direction 2

The current section is organized as follows. The first step (cf. Fig. 76) is automatically performed by the trained NLP algorithm (cf. Fig. 79) and it is presented in section 3.2.4.1, while the second step is manually performed and presented in section 3.2.4.2. Based on the conducted example, the achieved results concerning our purposes (cf. Table 9) are finally commented in section 3.2.4.3, also highlighting some important progresses embedded in the current demonstrator.

3.2.4.1. Step 1: identification of the SPs

The NLP algorithm schematized in Fig. 79 is characterised by five ANNs and five LTR databases (cf. Fig. 79), in agreement with the results of the training process (cf. Table 91 and Table 93), and it was implemented in MATLAB [48]. The intuitive user interface consists of a unique Excel table, which is shown in Fig. 99a. The data contained in the green cells are initialized by the user, and they involve the CFs of the input initial idea (cf. Fig. 98) and the input specifications (cf. Table 94). The resulting SPs are directly reported on the MATLAB command window with the same format as the SPs' database (specification, action, indication and eventual supporting DC, cf. Fig. 93), and they are shown in Fig.

99b. The supporting DCs extracted by the DCs' database (cf. Fig. 94) are shown in Fig. 100-Fig. 104, where the numbers correspond to the indices of the resulting SPs (cf. Fig. 99b).



Enter CFs		
Group Name	CF	Value
CFs^1	free	0.67
	linear	1
	close	0
CFs^2	straight	0
	extremity	1
CFs^3	wire	0
	spring	0
	truss	0
	beam	1
CFs^4	carriage	0
	pin	0
	joint	1
CFs^5	traction	0
	compression	0.5
	bending	1

Enter specs.	m_- U1_- U2_-

```
INDEX   SPEC.                ACTION   INDICATION    DC  FITNESS
   1     m_-                    A_-                          1
   2     m_-                  rho_-                          1
   3     m_-             variable_+                          1
   4     m_-             variable_+    bending_w           0.6
   5     m_-    close_truss_frame_+       beam_-   yes     0.8
   6    U1_-                    E_+                          1
   7    U1_-                   Jb_+                          1
   8    U1_-                   Jb_+    bending_w           0.8
   9    U1_-    truss_compression_+                yes     0.4
  10    U1_-    truss_compression_+        pin_+   yes       1
  11    U1_-       truss_traction_+        pin_+   yes     0.8
  12    U1_-        wire_traction_+        pin_+   yes     0.8
  13    U1_-    close_truss_frame_+                yes       1
  14    U1_-    close_truss_frame_+        pin_+   yes       1
  15    U2_-
```

Fig. 99: (a) Input user interface of the demonstrator; (b) SPs provided by the implemented NLP algorithm (cf. Fig. 79).

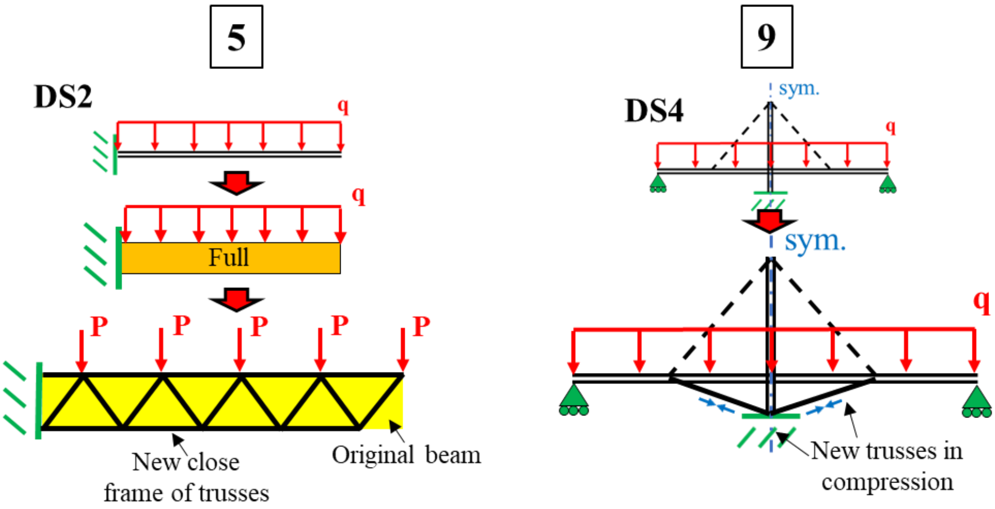


Fig. 100: DCs supporting the SPs of indices 5 and 9 (cf. Fig. 99b).

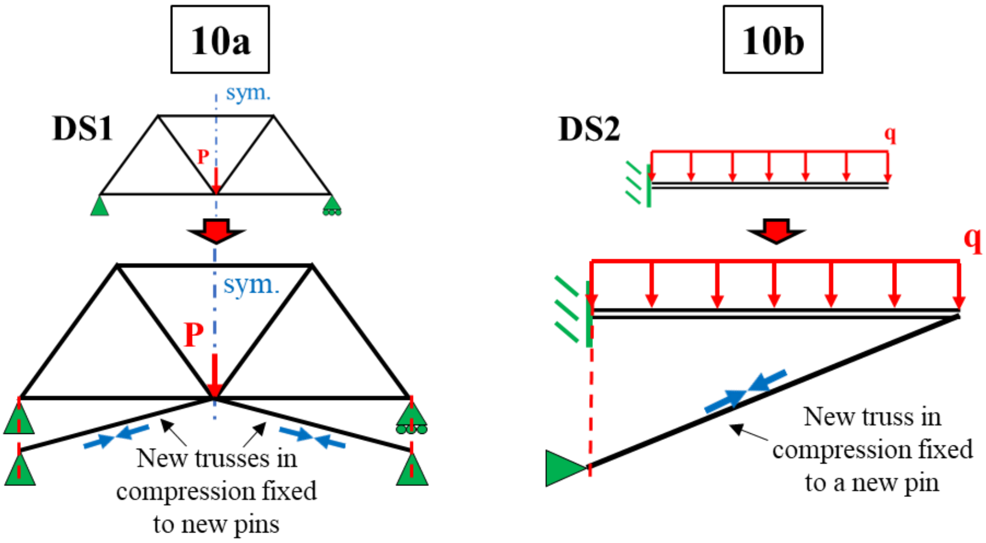


Fig. 101: DCs supporting the SP of index 10 (cf. Fig. 99b).

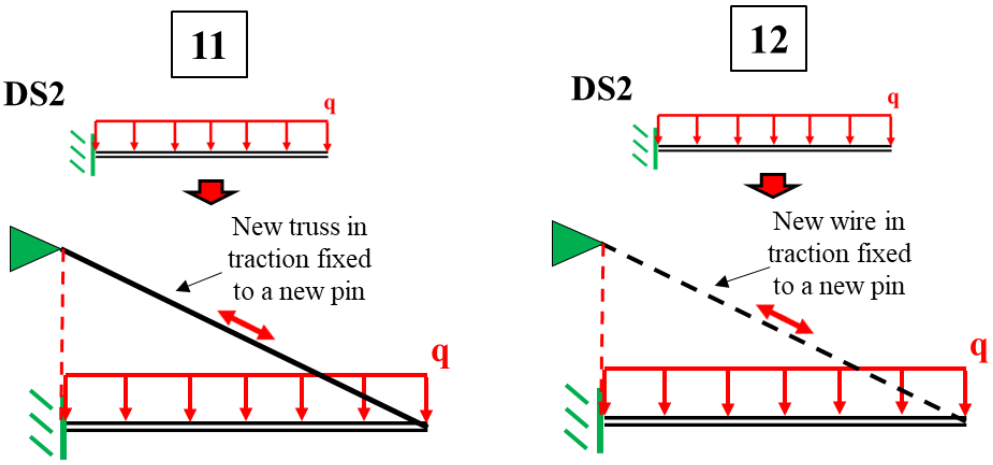

11
DS2
q
New truss in traction fixed to a new pin
q
12
DS2
q
New wire in traction fixed to a new pin
q

Fig. 102: DCs supporting the SPs of indices 11 and 12 (cf. Fig. 99b).

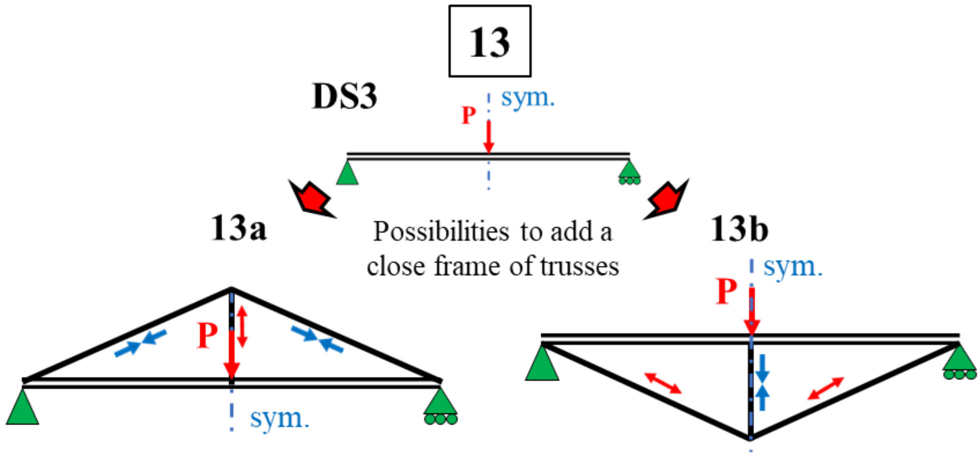


Fig. 103: DCs supporting the SP of index 13 (cf. Fig. 99b).

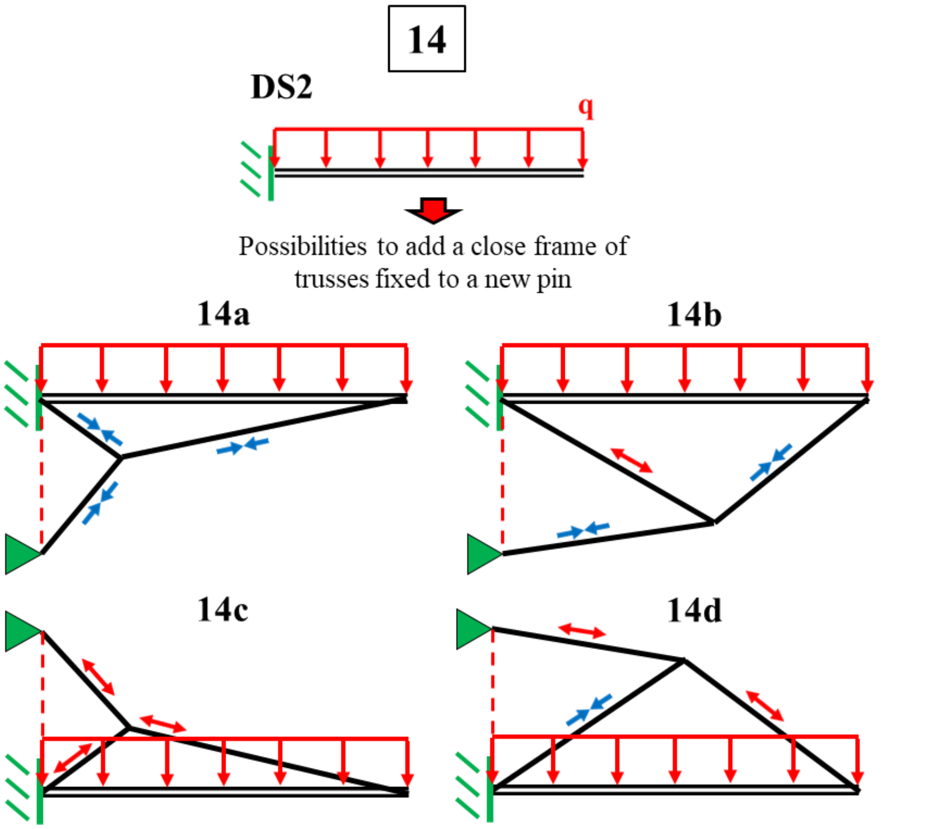

14
DS2
q
Possibilities to add a close frame of trusses fixed to a new pin
14a
14b
14c
14d

Fig. 104: DCs supporting the SP of index 14 (cf. Fig. 99b).

Observing Fig. 99b, we can see that multiple SPs were provided for the specifications "m_-" and U1_-", while no SP was provided for "U2_-". Some of the provided SPs can be considered equivalent in the present case, i.e. the SPs 3 and 4 and the SPs 7 and 8, respectively, since the overall structure of the initial idea is solicited to bending (cf. Fig. 98). Regarding "m_-" and U1_-", we obtained different SPs enabling to considerably expand the solution space and to act on the initial idea at different scales: material properties, section properties and structure topology (cf. Table 86). The evolution of the structure topology is particularly encouraged by the presence of many SPs of types 5 and 6 (cf. Table 89). Indeed, the respective supporting DCs provide many ideas to adopt different types of reinforcements based on wires and trusses (cf. Fig. 100-Fig. 104), but also to radically change the base structure. For example, the DC of index 5 (cf. Fig. 100) illustrates the replacement of a full beam with a frame of trusses. The fitness of the resulting SPs is on average high, between 0.8 and 1 (cf. Fig. 99b). According to these values, the majority of the SPs represent potentially relevant conceptual strategies[23], able to satisfy the corresponding specification. Finally, the NLP algorithm provided no SP for the specification "U2_-" (index 15, cf. Fig. 99b), while there effectively exists a non-zero displacement along the direction 2 in the initial idea (cf. Fig. 79). However, according to the provided DCs (cf. Fig. 100-Fig. 104), the majority of the resulting SPs is based on DS2 and DS3 (cf. Fig. 78), where U2 is zero due to the assumptions made in Table 83. It is thus important to remark that the absence of SPs for "U2_-" is due to the limited size of the knowledge database, which is based on very few DSs. In the following step of the demonstrator, we will focus on "m_-" and U1_-".

3.2.4.2. Step 2: development of the CVs

The development of the concept variants is manually conducted by applying the SPs provided by the NLP algorithm to the initial idea, and it is shown in Fig. 105. According to Fig. 76, this process is based on the interpretation of the SPs reported in Table 89, and on the analogies found between initial idea and supporting DCs (cf. Fig. 100-Fig. 104). As shown in Fig. 105, each SP is first applied to the initial idea following the order of the indices in Fig. 99b, and the applied SPs are then combined into new CVs. Based on the first hypothesis of the demonstrator (cf. Table 87), some of our memory biases could be introduced in both phases, and the numerical evaluation of the developed CVs is not performed: we remain at a conceptual level, where "everything" is potentially possible. According to our purposes (cf. Table 9), this enables to mainly focus on finding original ways to apply and combining the SPs, considering less important the technical details concerning the statics of the generated CVs[25]. For brevity, the following assumptions were made in Fig. 105:

- As anticipated in section 3.2.4.1, we focus on solving the specifications "m_-" and "U1_-", since no SP was for now provided for "U2_-" (cf. Fig. 99b).
- We mainly focus on the SPs supported by the DCs (i.e. 5, 9-14, cf. Fig. 99b), which enable to better visualize the evolution of the initial idea. It is assumed that the other SPs can be applied very easily and quickly by an expert user.
- A maximum of two application possibilities are given for each SP, and some SPs are considered in couple, as the SPs 9 and 10 and the SPs 11 and 12 (Fig. 99b), respectively, whose supporting DCs involve similar solutions.

[25] This aspect is object of the future developments and it is resumed in section 3.2.5.

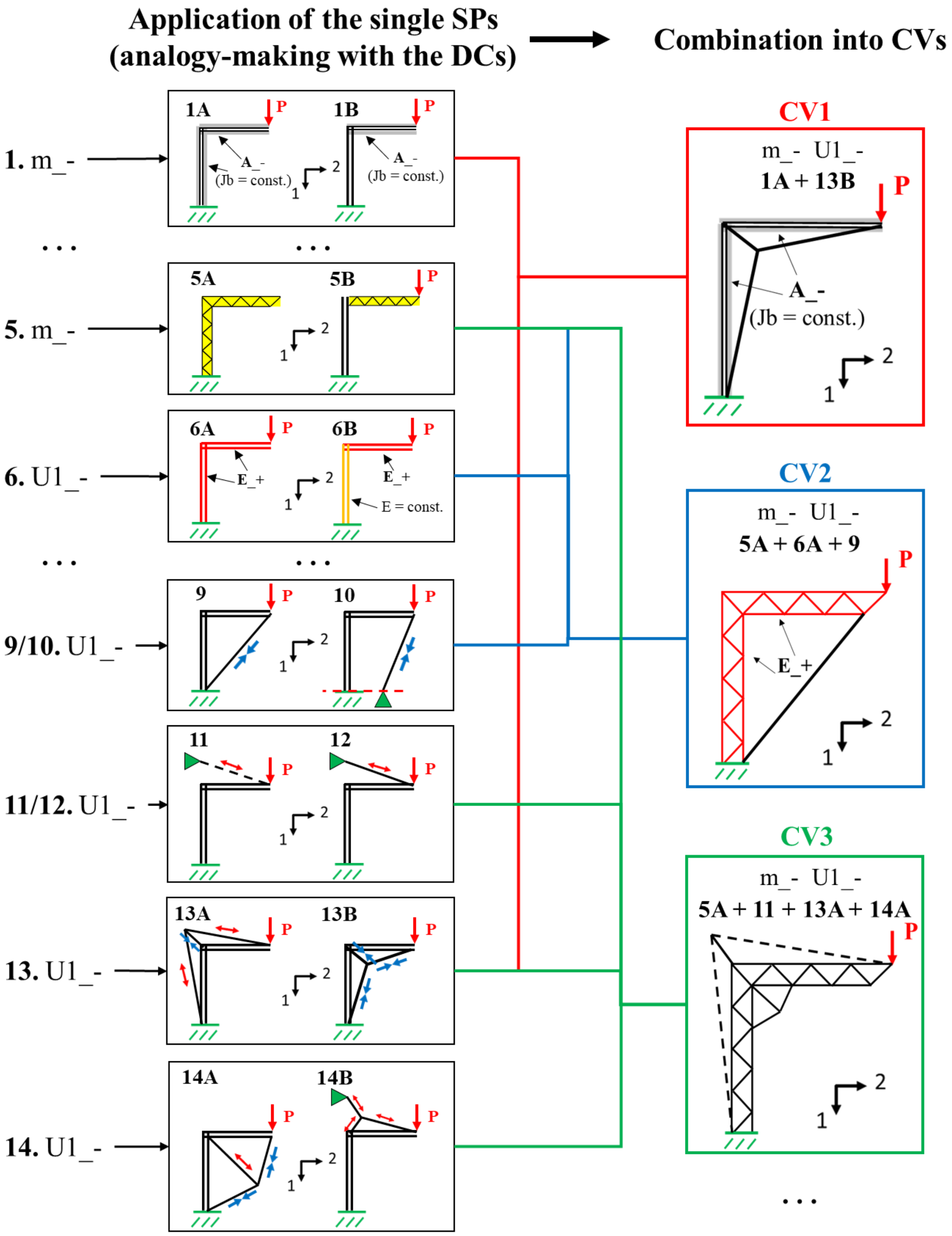


Fig. 105: Development of the CVs based on the application of the SPs to the initial idea (cf. Fig. 79 and Fig. 98).

The different SPs combined into the generated CVs were chosen arbitrarily, and the number of combined SPs increases going from CV1 (2 SPs combined) to CV3 (4 SPs combined). We are aware that our memory biases could have potentially influenced both the phases of analogy-making and combination (cf. Fig. 105). According to the required specifications, each CV shown in Fig. 105 includes conceptual solutions for answering to both "m_-" and U1_-". The topology of the CVs is considerably evolved from that of the initial idea. In accordance with the adopted approach (PAAs, cf. Table 18), all changes to the original structure were introduced without the help of any equation, but only by following the indications of the SPs, whose interpretation is reported in Table 89, and by finding analogies between the initial idea and the supporting DCs. According to Fig. 105, the most exploited analogy concerns the horizontal part of the initial idea and the design samples DS2 and DS3 (cf. Fig. 78), on which the majority of the supporting DCs is based (cf. Fig. 100-Fig. 104). We believe that all the generated CVs represent valid and relevant solutions, at least at a conceptual level. The effect of the combined SPs will be numerically weighted by the user in the subsequent embodiment design step (cf. Fig. 2), by transforming the CVs into design layouts to be tested and optimized.

3.2.4.3. Achieved results and important progresses concerning our purposes

Based on the presented example, the most important results concern the NLP algorithm used in the first step (cf. section 3.2.4.1). In agreement with the BDPs (cf. Table 5), the NLP algorithm provides multiple and different SPs (cf. Fig. 99b), pertinent to the input initial idea and enabling to act on the initial idea at different scales. This result is even more encouraging if we consider that the knowledge database of the NLP algorithm is based on a very limited number of DSs (cf. Table 87), and that the initial idea considered in the presented example is not included among them, i.e. it is unknown. Thanks to the non-numerical form of the SPs, there is no distinction between more performing ("good") and less performing ("bad") SPs, and this involves a minor focus on optimal solutions (cf. Table 9). Even with a very limited base lexicon (cf. Table 87), the defined SPs can express simple as well as more complex design concepts, which can be easily applied to the initial idea with the help of the DCs. All these elements contribute to enlarge the solution space, limiting the effects of cognitive biases (cf. Fig. 11) and providing the designer many different ideas for the conceptual development of the initial idea. Two other characteristics of the developed demonstrator represent an important progress towards the promotion of a design approach more oriented to improvisation and less depending on technical design rules (cf. Table 5 and Table 9):

- The implemented drawing tool (cf. section 3.2.2.2): it not only facilitates the definition of the CFs in the initial encoding step (cf. Fig. 79), but it also represents an original way to promote hand-sketching, whose benefits are summarised in Table 8.
- The process of analogy-making and SPs combination presented in Fig. 105 (cf. section 3.2.4.2), by which the user have more possibilities to invent and to take unconventional design choices

Both characteristics are object of future developments, and they will be resumed in section 3.2.5.

3.2.5. Future developments

The original conceptual design methodology presented in this chapter is the first demonstrator a future advanced software for aided breakthrough design. Based on the encouraging results achieved with the NLP algorithm and on the made progresses (cf. section 3.2.4.3), many future developments of the demonstrator are possible. The most important ones are reported as follows in order of increasing implementation difficulty:

- **Enlargement of the knowledge database** (cf. Fig. 92-Fig. 94). This first requires expanding the set of considered DCs (cf. Fig. 78) and the base lexicon (cf. Table 84-Table 86), then generating additional SPs and supporting DCs, possibly by also adopting new SP types (cf. Table 89). This development (ideally at short-term) would widen the potential of the NLP algorithm, which would be able to process many more different initial ideas and specifications and to propose many more diverse conceptual solutions in the first step.
- **Processing of "incomplete" initial ideas**. In agreement with the adopted approach (PAAs, cf. Table 18), both the drawing tool and the NLP algorithm are free from any model or equation regarding the statics and the mechanical response of the studied structure[26]. Thanks to this aspect and to the use of the masking word technique in the generation of the calibration set (cf. Fig. 97 and Table 71), the drawing tool and NLP algorithm are potentially able to process "incomplete" initial idea, i.e. characterised by "missing information", as for example a non-assigned element type or a missing constraint which makes the structure unstable. First, is this feature useful? We believe so: it is possible that the user of the demonstrator may have an unclear initial idea (and thus incomplete), especially at the beginning of the conceptual design. Therefore, how much incomplete can the initial idea be for obtaining relevant results with the NLP algorithm? Answering to the latter question requires further specific tests, whose development represents a future middle-term objective.
- **Performing the second step with the machine**. The second step of the conceptual design procedure is for now manually conducted by the user (cf. Fig. 76 and section 3.2.4.2). As said in section 3.2.4.3, we believe that the hypotheses made in Table 87 favour the outflow of the user's spirit of invention and a minor focus on technical details during the generation of the CVs, in accordance with our purposes (cf. Table 9). However, the user is hopelessly conditioned by his memory biases. As experts, we will mostly be influenced by our subconscious functional fixedness (cf. Table 6), which will always prevent us to "think outside the box", more or less significantly. The implementation of an algorithm emulating the processes of analogy-making and of SPs' combination illustrated in Fig. 105 represents the most challenging (and thus a long-term) future development. In an even more distant but not utopistic future, such algorithm will also be able to autonomously test the generated CVs and to automatically feed the knowledge database of the NLP algorithm, thus leading to an extremely powerful conceptual design software.

[26] The FEM models used to test the efficacy of the generated SPs represent a pure verification tool exterior to the NLP methodology, and they a0re not used in the demonstrator to provide responses (cf. section 3.2.3.1).

Conclusions

Conclusions are afterwards presented by retracing the main points of our work and by highlighting the original contributions. The focus, the purposes and the approach pursued in the current thesis were outlined chapter 1 by following the original path illustrated in Fig. 1. The focus of the work was immediately clarified in section 1.1, where we first introduced the three main roles of design (cf. Table 1) and a well-established model of design process (cf. Fig. 2). We thus described each step. The conceptual design step was presented in Fig. 3, where the important notions of solution principle (SP) and concept variant (CV) are also introduced. We finally highlighted the fundamental differences between conceptual design, where "everything is possible", and the subsequent embodiment design, where the CVs are transformed in technical layouts and where everything is now constrained to the chosen preliminary conditions (cf. Fig. 4).

The definition of the purposes (cf. section 1.2) was conducted in an original way, starting by analysing the evolution of the innovation concept in the industrial and entrepreneurial worlds in the second half of the 20th century, and its relationship with design. The adopted definition of innovation is reported in Fig. 7, which highlights the three principal aspects of novelty (diversity), combination and change. Following the path in Fig. 1, we then focused on breakthrough innovations, whose key element is represented by the concept originality (cf. section 1.2.2.2), and we formulated three important breakthrough design practices (BDPs, cf. Table 5) based on the human tacit knowledge (cf. Table 3). As remarked in section 1.2.2.7, the BDPs cannot guarantee the achievement of breakthrough innovations but, if observed, they enable to enormously enlarge the solution space, increasing the possibility to include and reach breakthrough design solutions. The subsequent analysis of the cognitive biases revealed that BDPs are usually ignored by designers, who tend instead to subconsciously follow some "bad" design habitudes (BDHs, cf. Table 7). After studying the role of cognitive biases in inducing the BDHs and in restricting the solution space (cf. Fig. 11), we introduced some important principles to mitigate their influence on the design process (cf. Table 8). Based on the BDPs and on the study of the cognitive biases, we finally formulated three purposes (cf. Table 9) aimed to favour the birth of original design concepts (i.e. the key to breakthrough innovations, cf. section 1.2.2.2), and which were afterwards pursued in the development of our demonstrator.

The approach followed to implement the demonstrator was presented in section 1.3 in an original way. We first clarified that optimization, i.e. the most widespread design approach, is out of our scope. This was done by performing an optimization example (cf. sections 1.3.1.2 and 1.3.1.3), and by identifying the aspects of this approach which collide with our purposes (Table 11). Many of these aspects can be reconducted to the BDHs (cf. Table 7), as the excessive focus on technical details. To better introduce our approach, we then studied a well-established conceptual design theory, i.e. TIPS, based on the use of equation-free design methods and of natural language. The analysis of an application example of TIPS reported in literature shown that these features involve a minor focus on technical aspects and optimal solutions during the design process, enabling to consider different types of solutions to enlarge the research space, in agreement with the formulated purposes (cf. Table 9). Based on these ideas, we finally defined the principles of the adopted approach (cf. Table 18), which were afterwards followed during the implementation of the demonstrator.

The developed machine learning methodology was presented in chapter 2 by following the original path in Fig. 23. We first explained the reasons for adopting machine learning in section 2.1, by comparing human brain to machine. We made a preliminary overview on some basic biological aspects of the brain (neurons and synapses), and on memory, analysing how information is processed, stored and retrieved (cf. Fig. 25). Thanks to this study, we identified some aspects of our memory which could represent a limit during conceptual design (cf. Table 22). An original interpretation of the designer's mental processes was assumed in Fig. 28 and to introduce our three main key cognitive abilities (abstraction, generalization and analogy-making), and to explain their roles in conceptual design (cf. Table 24). As assumed in Fig. 28, the power of the KCAs enables our brain to quickly retrieve the design concepts previously interiorized, to manipulate them in a dialectic form (natural language) and to adapt them by means of similitudes, without executing any equation. On the other hand, the original brain-machine comparison conducted in Table 25 revealed that our thinking is constantly influenced by the unconscious mechanisms due to memory which are almost impossible to eliminate (cf. Table 22), and which are dangerous to our purposes. Moreover, our brain is inherently limited in language variety and multidisciplinary knowledge, whereas the machine has potentially unlimited capabilities. It is not excluded that cognitive biases can be introduced in knowledge base of the machine by programmer's choices, but their effects can be mitigated by increasing the number of programmers. Based on the presented elements, 0the idea of the developing a machine learning methodology was to emulate the KCAs, exploiting their power and all the advantages offered by machine during conceptual design, without being affected by brain limits (cf. Table 25). Due to this ambitious idea, in this work we focused our efforts on the KCAs of abstraction and generalization (step 1, Fig. 28). This assumption was also reported in Fig. 23.

In section 2.2, we presented the developed ML methodology. The three principal ML models were first analysed, and their main characteristics were summarised in Table 28. Among them, the supervised and unsupervised models were judged as more suitable, and thus adopted, to emulate the KCAs of abstraction and generalization, respectively. Based on the theory provided in the literature, we implemented a versatile model of artificial neural network (cf. Fig. 34) to perform supervised learning, and a genetic algorithm (cf. Fig. 51) to perform the unsupervised learning by means of the method K-means (cf. Fig. 46). Both algorithms were implemented in MATLAB [48], and their set-up is entirely handled with intuitive Excel interfaces (cf. Fig. 39 and Fig. 56), which represent an original contribution of this work. For both algorithms, we analysed the role of the main hyperparameters (cf. Table 34 for ANNs and Table 57 for GA), and we conducted different numerical tests. The implemented ANN was tested with an original example of image recognition (cf. Fig. 40), while the response of our GA was evaluated in a partitional clustering problem (cf. Fig. 57). Numerical tests finally confirmed the robustness and the relevancy of the results provided by both algorithms.

Chapter 3 was devoted to natural language processing and to the implementation of the demonstrator, following the path in Fig. 60. In section 3.1, the ML methods developed in chapter 2 were combined into an original methodology for the processing of every-day language. Based on the ideas introduced in Fig. 61, the NLP algorithm (cf. Fig. 68) analyses the context of an incomplete input sentence via an ANN (cf. Fig. 67b), and it then proposes multiple coherent continuations exploiting the LTR models obtained by the training sentences (cf. Fig. 65). The NLP algorithm was implemented in MATLAB [48], and some aspects of its training are inspired to the BERT algorithm of Google [111], as the two steps of pre-training and fine-training (cf. Fig. 62) and the use of the masking word technique (cf. Fig. 67a and Table 71). Preliminary tests (cf. Fig. 72) revealed that the NLP algorithm is quite robust in processing unknown sentences and in proposing different coherent continuations. This result is very encouraging, mostly considering the very limited size of the training set (50 sentences and 133 different words, cf. Table 72), whereas a 6-years-old child, for example, already knows 6,000 different words [117]. The very limiting aspect concerned the reduced number of proposed continuations (examples 2 and 3, cf. Fig. 72b), which was however improved in an original way. Instead of considering a unique group of CFs, these were divided into two groups (cf. Table 78), from which two distinct LTR databases and two distinct ANNs were derived (cf. Fig. 73 and Table 82, respectively). In the improved NLP algorithm, the two designated LTR models are thus combined into a unified context (cf. Fig. 74). Based on the results in Fig. 75, this improvement enables to increase the number of proposed continuations without enlarging the initial set of training sentences, despite it extends the duration of the training process (cf. 3.1.3.3).

The original demonstrator of conceptual design procedure was finally presented in section 3.2. We immediately gave the two principal hypotheses (cf. Table 87). Based on Fig. 28, the first hypothesis outlines the roles of the machine (NLP algorithm) and of the user (expert engineer) in the conceptual design steps, providing the scheme of the demonstrator (cf. Fig. 76). The second hypothesis concerns the knowledge base of the machine, which is limited to 5 design samples (DSs) and less than 30 different words (cf. Fig. 77), and it included all the assumptions about the considered DSs (cf. Fig. 78 and Table 83). The idea is to show that the machine is able to express complex design concepts and to provide relevant results with few words and few design examples to which refer. The NLP algorithm used in the first step (cf. Fig. 79) is based on the improved NLP methodology (cf. Fig. 74), whose adaptation to conceptual design was described as follows. We first illustrated the criteria to define the values of the CFs (cf. Table 88), which represent an original and effective way of describing a mechanical structure with the natural language. We thus implemented an original and intuitive drawing tool (cf. Fig. 86) which facilitates the definition of the CFs' values by the user and which promotes hand-sketching (cf. Table 8), with the idea to mitigate cognitive biases according to our purposes (cf. Table 9). Finally, we defined the architecture and the types of the SPs (cf. Fig. 90 and Table 89), which are characterised by an original abbreviated form facilitating the construction of the LTR models (cf. Fig. 79). The afterwards training of the NLP algorithm was conducted as shown in Fig. 91. In the training set-up, the knowledge database was organized in three distinct levels: CFs' values of the five initial DSs considered in Fig. 78 (level 1, cf. Fig. 92), database of the SPs (level 2, cf. Fig. 93), and morphological matrix of the supporting DSs (level 3, cf. Fig. 93). They were initialized in Excel datasheets to facilitate future developments of the knowledge database. The training process of the NLP algorithm was then conducted as in the improved NLP methodology (cf. section 3.1.3.1). After training the NLP algorithm, the demonstrator was tested by adopting the initial idea and the specifications reported in Fig. 79. The CFs' values were determined with the implemented drawing tool (cf. Fig. 98), and the SPs proposed by the NLP algorithm in the first step were reported in Fig. 99b. In general, these original results show that the machine provides different SPs enabling to considerably expand the solution space, in agreement with our purposes. This is very encouraging as the knowledge of the NLP algorithm is based on few words and on few DSs (cf. Table 87), which however implies that some specifications could be unanswered (as "U2_-", cf. Fig. 99b). The supporting DCs (cf. Fig. 100-Fig. 104) provide the user many ideas to adopt different types of reinforcements and to radically evolve the initial idea. This aspect of the demonstrator is very useful mostly in the second step (cf. Fig. 105), which was for now manually conducted according to the made hypotheses (cf. Table 87). The most important results and progresses concerning our purposes were discussed in section 3.2.4.3. At the end of chapter 3, we proposed three possible future development of the demonstrator (cf. section 3.2.5), highlighting the will to pursue our work for achieving a complete and more powerful conceptual design software.

References

1. Fleming L. Breakthroughs and the ‘Long Tail’ of Innovation. *MIT Sloan Managment Review* 2007; **49**(1).
2. Mascitelli R. From Experience: Harnessing Tacit Knowledge to Achieve Breakthrough Innovation. *Journal of Product Innovation Management* 2000; **17**(3): 179–193. DOI: 10.1111/1540-5885.1730179.
3. Liedtka J. Perspective: Linking Design Thinking with Innovation Outcomes through Cognitive Bias Reduction. *Journal of Product Innovation Management* 2014; **32**(6): 925–938. DOI: 10.1111/jpim.12163.
4. Brockis J. Overcoming cognitive bias is a smart way to foster innovation. *Australian Law Management Journal* 2017.
5. Brockis J. *Future Brain: The 12 Keys to Create Your High-Performance Brain*. Wiley; 2015.
6. Bont C, Liu S. Breakthrough Innovation through Design Education: Perspectives of Design-Led Innovators. *Design Issues* 2017; **33**: 18–30. DOI: 10.1162/DESI_a_00437.
7. White A. *From Comfort Zone to Performance Management: Understanding development and performance*. Belgium: White & MacLean; 2009.
8. Leon N, Gutierrez J, Martinez O, Castillo C. Optimization VS Innovation in a Cae Environment. In: Jacquart R, editor. *Building the Information Society*, Boston, MA: Springer US; 2004. DOI: 10.1007/978-1-4020-8157-6_43.
9. Dubois S, Rasovska I, De Guio R. Comparison of non solvable problem solving principles issued from CSP and TRIZ. *IFIP International Federation for Information Processing* 2008; **277**. DOI: 10.1007/978-0-387-09697-1_7.
10. Hernández RJ, Cooper R, Tether B, Murphy E. Design, the Language of Innovation: A Review of the Design Studies Literature. *She Ji: The Journal of Design, Economics, and Innovation* 2018; **4**(3): 249–274. DOI: 10.1016/j.sheji.2018.06.001.
11. Malmqvist J, Axelsson R, Johansson M. A Comparative Analysis of the Theory of Inventive Problem Solving and the Systematic Approach of Pahl and Beitz, *The 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*. Irvine, California: 1996.
12. Pannozzo A. The (Ir) relevance of Technology: Creating a Culture of Opportunity by Design. *Design Management Review* 2007; **18**(4): 18–24. DOI: 10.1111/j.1948-7169.2007.tb00090.x.
13. Godin B. “Innovation Studies”: The Invention of a Specialty. *Minerva* 2012; **50**(4): 397–421. DOI: 10.1007/s11024-012-9212-8.
14. Hobday M, Boddington A, Grantham A. An Innovation Perspective on Design: Part 1. *Design Issues* 2011; **27**(4): 5–15. DOI: 10.1162/DESI_a_00101.
15. Herbert A. S. *The Sciences of the Artificial*. 3rd edition. Cambridge, MA: MIT Press; 1996.
16. Pahl G, Beitz W, Feldhusen J, Grote KH. *Engineering Design: a Systematic Approach*. 3rd ed. London: Springer-Verlag; 2007.
17. Gassmann O, Zeschky M. Opening up the Solution Space: The Role of Analogical Thinking for Breakthrough Product Innovation. *Creativity and Innovation Management* 2008; **17**(2): 97–106. DOI: 10.1111/j.1467-8691.2008.00475.x.
18. Mayda M, Börklü HR. An integration of TRIZ and the systematic approach of Pahl and Beitz for innovative conceptual design process. *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 2014; **36**(4): 859–870. DOI: 10.1007/s40430-013-0106-y.
19. Fagerberg J, Verspagen B. Innovation studies—The emerging structure of a new scientific field. *Research Policy* 2009; **38**(2): 218–233. DOI: 10.1016/j.respol.2008.12.006.
20. Śledzik K. Schumpeter’s View on Innovation and Entrepreneurship. *SSRN Electronic Journal* 2013. DOI: 10.2139/ssrn.2257783.
21. Freeman C. *The Economics of Industrial Innovation*. Penguin; 1974.
22. Belcher A. The Invention, Innovation and Diffusion of Self-Regulation in Corporate Governance. *Northern Ireland Legal Quarterly* 1996; **47**(3): 322–334.

23. Freeman C, Young MA, Fuller J. The plastic industry: a comparative study of research and innovation. *National Institute Economic Review* 1963(26): 22–62.

24. OECD. *Frascati Manual 2015: Guidelines for Collecting and Reporting Data on Research and Experimental Development*. Paris: OECD Publishing; 2015.

25. OECD. *Government and technical innovation.* Paris: OECD; 1966.

26. OECD. *Gaps in technology: Comparisons between member countries in education, R&D, technological innovation*. Paris: International Economic Exchanges; 1970.

27. OECD/Eurostat. *Oslo Manual 2018: Guidelines for Collecting, Reporting and Using Data on Innovation*. 4th Edition. Paris/Eurostat, Luxembourg: OECD Publishing; 2019.

28. Chandy RK, Tellis GJ. Organizing for Radical Product Innovation: The Overlooked Role of Willingness to Cannibalize. *Journal of Marketing Research* 1998; **35**(4): 474–487. DOI: 10.2307/3152166.

29. Polanyi M. *The Tacit Dimension*. 1st Edition. Doubleday; 1966.

30. Fleming L. Recombinant Uncertainty in Technological Search. *Management Science* 2001; **47**(1): 117–132. DOI: 10.1287/mnsc.47.1.117.10671.

31. Loh P. *Achieving breakthroughs in Innovation*. Knowledgeworks Consultants; 2008.

32. de Vere I. Developing creative engineers: a design approach to engineering education, *International Conference on Engineering and Product Design Education*. 2009.

33. Lande M, Leifer L. Difficulties Student Engineers Face Designing the Future. *International Journal of Engineering Education* 2010; **26**(2).

34. Ariely D. *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. Canada: HarperCollins; 2008.

35. Santamarina JC. Creativity and Engineering, Education Strategies, *International Conference/Workshop on Engineering Education Honouring Professor James T.P. Yao*. Texas: 2002.

36. Changqing G, Kezheng H, Fei M. Comparison of innovation methodologies and TRIZ. *The TRIZ Journal* 2005: 1–5.

37. Harryson SJ. From experience How Canon and Sony drive product innovation through networking and application-focused R&D. *Journal of Product Innovation Management* 1997; **14**(4): 288–295. DOI: 10.1016/S0737-6782(97)00011-8.

38. Goel V. *Sketches of Thought*. Cambridge, MA, USA: MIT Press; 1995.

39. Ullman DG, Wood S, Craig D. The importance of drawing in the mechanical design process. *Computers & Graphics* 1990; **14**(2): 263–274. DOI: 10.1016/0097-8493(90)90037-X.

40. Deb K. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley; 2001.

41. Gen M, Cheng R. *Genetic Algorithms and Engineering Design*. John Wiley & Sons; 1996.

42. Nunziante L, Gambarotta L, Tralli A. *Scienza delle costruzioni*. 3rd ed. McGraw-Hill; 2011.

43. Altshuller G, Altov H. *And Suddenly the Inventor Appeared: TRIZ, the Theory of Inventive Problem Solving*. Technical Innovation Center, Inc.; 1996.

44. Altshuller. *Creativity As an Exact Science*. CRC Press; 1984.

45. Mayda M, Borklu HR. Development of an innovative conceptual design process by using Pahl and Beitz's systematic design, TRIZ and QFD. *Journal of Advanced Mechanical Design, Systems, and Manufacturing* 2014; **8**(3): JAMDSM0031–JAMDSM0031. DOI: 10.1299/jamdsm.2014jamdsm0031.

46. Rowan J. *Ordinary Ecstasy: The Dialectics of Humanistic Psychology*. 3rd ed. Taylor & Francis Ltd; 2001.

47. El Naqa I, Murphy MJ. What Is Machine Learning? In: El Naqa I, Li R, Murphy MJ, editors. *Machine Learning in Radiation Oncology: Theory and Applications*, Cham: Springer International Publishing; 2015. DOI: 10.1007/978-3-319-18305-3_1.

48. MathWorks. *MATLAB R2020b*. Natick, Massachusetts, United States: 2020.

49. Kalita H, Krishnaprasad A, Choudhary N, Das S, Dev D, Ding Y, *et al.* Artificial Neuron using Vertical MoS 2 /Graphene Threshold Switching Memristors. *Scientific Reports* 2019; **9**(1): 53. DOI: 10.1038/s41598-018-35828-z.

50. Kriesel D. *A brief introduction to neural networks*. 2nd edition. dkriesel.com; 2007.

51. Südhof TC, Malenka RC. Understanding Synapses: Past, Present, and Future. *Neuron* 2008; **60**(3): 469–476. DOI: 10.1016/j.neuron.2008.10.011.

52. Atkinson RC, Shiffrin RM. Human Memory: A Proposed System and its Control Processes. In: Spence KW, Spence JT, editors. *Psychology of Learning and Motivation*, vol. 2, Academic Press; 1968. DOI: 10.1016/S0079-7421(08)60422-3.

53. Malmberg KJ, Raaijmakers JGW, Shiffrin RM. 50 years of research sparked by Atkinson and Shiffrin (1968). *Memory & Cognition* 2019; **47**(4): 561–574. DOI: 10.3758/s13421-019-00896-7.

54. Shallice T, Warrington EK. Auditory-verbal short-term memory impairment and conduction aphasia. *Brain and Language* 1977; **4**(4): 479–491. DOI: 10.1016/0093-934X(77)90040-2.

55. Glanzer M. Distance between related words in free recall: Trace of the STS. *Journal of Verbal Learning and Verbal Behavior* 1969; **8**(1): 105–111. DOI: 10.1016/S0022-5371(69)80018-6.

56. Cowan N. Chapter 20 What are the differences between long-term, short-term, and working memory? In: Sossin WS, Lacaille JC, Castellucci VF, Belleville S, editors. *Progress in Brain Research*, vol. 169, Elsevier; 2008. DOI: 10.1016/S0079-6123(07)00020-9.

57. Norris D, Kalm K, Hall J. Chunking and Redintegration in Verbal Short-Term Memory. *Journal of Experimental Psychology Learning, Memory, and Cognition* 2020; **46**(5): 872–893. DOI: 10.1037/xlm0000762.

58. Mattarella-Micke A, Beilock SL. Capacity Limitations of Memory and Learning. In: Seel NM, editor. *Encyclopedia of the Sciences of Learning*, Boston, MA: Springer US; 2012. DOI: 10.1007/978-1-4419-1428-6_603.

59. Chai WJ, Abd Hamid AI, Abdullah JM. Working Memory From the Psychological and Neurosciences Perspectives: A Review. *Frontiers in Psychology* 2018; **9**. DOI: 10.3389/fpsyg.2018.00401.

60. Bailey CH, Bartsch D, Kandel ER. Toward a molecular definition of long-term memory storage. *Proceedings of the National Academy of Sciences* 1996; **93**(24): 13445–13452.

61. Legrenzi P, Papagno C, Umiltà C. *Psicologia generale. Dal cervello alla mente*. Il Mulino; 2012.

62. Tulving E. *Elements of Episodic Memory*. Oxford University Press; 1983.

63. Tulving E, Donaldson W. *Organization of memory*. Oxford, England: Academic Press; 1972.

64. Tulving E. Episodic Memory and Autonoesis: Uniquely Human? *The missing link in cognition: Origins of self-reflective consciousness*, New York, NY, US: Oxford University Press; 2005. DOI: 10.1093/acprof:oso/9780195161564.003.0001.

65. Squire LR, Knowlton B, Musen G. The Structure and Organization of Memory. *Annual Review of Psychology* 1993; **44**(1): 453–495. DOI: 10.1146/annurev.ps.44.020193.002321.

66. Squire LR. Declarative and Nondeclarative Memory: Multiple Brain Systems Supporting Learning and Memory. *Journal of Cognitive Neuroscience* 1992; **4**(3): 232–243. DOI: 10.1162/jocn.1992.4.3.232.

67. Nader K, Hardt O. A single standard for memory: the case for reconsolidation. *Nature Reviews Neuroscience* 2009; **10**(3): 224–234. DOI: 10.1038/nrn2590.

68. Abraham WC, Jones OD, Glanzman DL. Is plasticity of synapses the mechanism of long-term memory storage? *Npj Science of Learning* 2019; **4**(1): 1–10. DOI: 10.1038/s41539-019-0048-y.

69. Aquilar F, Pugliese MP. *Condividere i ricordi: psicoterapia cognitiva e funzioni della memoria*. Franco Angeli.; 2017.

70. Wang Y, Fariello G. On Neuroinformatics: Mathematical Models of Neuroscience and Neurocomputing. *Journal of Advanced Mathematics and Applications* 2012; **1**(2): 206–217. DOI: 10.1166/jama.2012.1015.

71. Wierzba M, Riegel M, Wypych M, Jednoróg K, Grabowska A, Marchewka A. Cognitive control over memory – individual differences in memory performance for emotional and neutral material. *Scientific Reports* 2018; **8**(1): 3808. DOI: 10.1038/s41598-018-21857-1.

72. Arnold MG. Combining conscious and unconscious knowledge within human-machine-interfaces to foster sustainability with decision-making concerning production processes. *Journal of Cleaner Production* 2018; **179**: 581–592. DOI: 10.1016/j.jclepro.2018.01.070.

73. Smith ER, DeCoster J. Dual-Process Models in Social and Cognitive Psychology: Conceptual Integration and Links to Underlying Memory Systems. *Personality and Social Psychology Review* 2000; **4**(2): 108–131. DOI: 10.1207/S15327957PSPR0402_01.

74. Amodio DM, Ratner KG. A Memory Systems Model of Implicit Social Cognition. *Current Directions in Psychological Science* 2011; **20**(3): 143–148. DOI: 10.1177/0963721411408562.

75. Mitchell M. Abstraction and Analogy-Making in Artificial Intelligence. *ArXiv:210210717 [Cs]* 2021.

76. French RM. The computational modeling of analogy-making. *Trends in Cognitive Sciences* 2002; **6**(5): 200–205. DOI: 10.1016/S1364-6613(02)01882-X.

77. Koziołek S. Design by Analogy: Synectics and Knowledge Acquisition Network. In: Rusiński E, Pietrusiak D, editors. *Proceedings of the 13th International Scientific Conference*, Cham: Springer International Publishing; 2017. DOI: 10.1007/978-3-319-50938-9_27.

78. Gassmann O, Zeschky M. Opening up the Solution Space: The Role of Analogical Thinking for Breakthrough Product Innovation. *Creativity and Innovation Management* 2008; **17**(2): 97–106. DOI: 10.1111/j.1467-8691.2008.00475.x.

79. Schmitt N, Cobb T, Horst M, Schmitt D. How much vocabulary is needed to use English? Replication of van Zeeland & Schmitt (2012), Nation (2006) and Cobb (2007). *Language Teaching* 2017; **50**(2): 212–226. DOI: 10.1017/S0261444815000075.

80. Brysbaert M, Stevens M, Mandera P, Keuleers E. How Many Words Do We Know? Practical Estimates of Vocabulary Size Dependent on Word Definition, the Degree of Language Input and the Participant's Age. *Frontiers in Psychology* 2016; **7**: 1116. DOI: 10.3389/fpsyg.2016.01116.

81. D'Anna CA, Zechmeister EB, Hall JW. Toward a Meaningful Definition of Vocabulary Size. *Journal of Reading Behavior* 1991; **23**(1): 109–122. DOI: 10.1080/10862969109547729.

82. Alloghani M, Al-Jumeily D, Mustafina J, Hussain A, Aljaaf AJ. A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In: Berry MW, Mohamed A, Yap BW, editors. *Supervised and Unsupervised Learning for Data Science*, Cham: Springer International Publishing; 2020. DOI: 10.1007/978-3-030-22475-2_1.

83. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA, USA: A Bradford Book; 2018.

84. Zhou ZH. A brief introduction to weakly supervised learning. *National Science Review* 2018; **5**(1): 44–53. DOI: 10.1093/nsr/nwx106.

85. Torrado RR, Bontrager P, Togelius J, Liu J, Perez-Liebana D. Deep Reinforcement Learning for General Video Game AI. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. 2018. DOI: 10.1109/CIG.2018.8490422.

86. Ahmad MW, Mourshed M, Rezgui Y. Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption. *Energy and Buildings* 2017; **147**: 77–89. DOI: 10.1016/j.enbuild.2017.04.038.

87. Shichkin AV, Buevich AG, Sergeev AP. Comparison of artificial neural network, random forest and random perceptron forest for forecasting the spatial impurity distribution. *AIP Conference Proceedings* 2018; **1982**(1): 020005. DOI: 10.1063/1.5045411.

88. Raczko E, Zagajewski B. Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images. *European Journal of Remote Sensing* 2017; **50**(1): 144–154. DOI: 10.1080/22797254.2017.1299557.

89. Han T, Jiang D, Zhao Q, Wang L, Yin K. Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control* 2018; **40**(8): 2681–2693. DOI: 10.1177/0142331217708242.

90. Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD. An introduction to decision tree modeling. *Journal of Chemometrics* 2004; **18**(6): 275–285. DOI: 10.1002/cem.873.

91. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb. *Journal of Electronic Science and Technology* 2019; **17**(1): 26–40. DOI: 10.11989/JEST.1674-862X.80904120.

92. Feurer M, Hutter F. Hyperparameter Optimization. In: Hutter F, Kotthoff L, Vanschoren J, editors. *Automated Machine Learning: Methods, Systems, Challenges*, Cham: Springer International Publishing; 2019. DOI: 10.1007/978-3-030-05318-5_1.

93. Koutsoukas A, Monaghan KJ, Li X, Huan J. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics* 2017; **9**: 42. DOI: 10.1186/s13321-017-0226-y.

94. Yuan B. Efficient hardware architecture of softmax layer in deep neural network. *2016 29th IEEE International System-on-Chip Conference (SOCC)*, *2016 29th IEEE International System-on-Chip Conference (SOCC)*. 2016. DOI: 10.1109/SOCC.2016.7905501.

95. Leonard J, Kramer MA. Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering* 1990; **14**(3): 337–341. DOI: 10.1016/0098-1354(90)87070-6.

96. Li J, Cheng J hang, Shi J yuan, Huang F. Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. In: Jin D, Lin S, editors. *Advances in Computer Science and Information Engineering*, Berlin, Heidelberg: Springer; 2012. DOI: 10.1007/978-3-642-30223-7_87.

97. Smith LN. A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay. *ArXiv:180309820 [Cs, Stat]* 2018.

98. Yu CC, Liu BD. A backpropagation algorithm with adaptive learning rate and momentum coefficient. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, vol. 2, *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. 2002. DOI: 10.1109/IJCNN.2002.1007668.

99. Qian N. On the momentum term in gradient descent learning algorithms. *Neural Networks* 1999; **12**(1): 145–151. DOI: 10.1016/S0893-6080(98)00116-6.

100. Yu XH, Chen GA. Efficient Backpropagation Learning Using Optimal Learning Rate and Momentum. *Neural Networks* 1997; **10**(3): 517–527. DOI: 10.1016/S0893-6080(96)00102-5.

101. Rani Y, Rohil H. A Study of Hierarchical Clustering Algorithm. *International Journal of Information and Computation Technology* 2013; **3**(10): 1115–1122.

102. Reynolds AP, Richards G, de la Iglesia B, Rayward-Smith VJ. Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms. *Journal of Mathematical Modelling and Algorithms* 2006; **5**(4): 475–504. DOI: 10.1007/s10852-005-9022-1.

103. Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery* 2012; **2**(1): 86–97. DOI: 10.1002/widm.53.

104. Kantardzic M. *Data Mining: Concepts, Models, Methods, and Algorithms*. 3rd ed. Wiley-IEEE Press; 2019.

105. Han J, Kamber M, Pei J. *Data Mining: Concepts and Techniques*. 3rd ed. Elsevier; 2012.

106. Arthur D, Manthey B, Röglin H. Smoothed Analysis of the k-Means Method. *Journal of the ACM* 2011; **58**(5): 19:1-19:31. DOI: 10.1145/2027216.2027217.

107. Yang XS. Chapter 6 - Genetic Algorithms. In: Yang XS, editor. *Nature-Inspired Optimization Algorithms (Second Edition)*, Academic Press; 2021. DOI: 10.1016/B978-0-12-821986-7.00013-5.

108. Gen M, Cheng R. *Genetic Algorithms and Engineering Design*. John Wiley & Sons; 1996.

109. Indurkhya N, Damerau FJ. *Handbook of Natural Language Processing*. 2nd edition. CRC Press; 2010.

110. Nadkarni PM, Ohno-Machado L, Chapman WW. Natural language processing: an introduction. *Journal of the American Medical Informatics Association* 2011; **18**(5): 544–551. DOI: 10.1136/amiajnl-2011-000464.

111. Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv:181004805 [Cs]* 2019.

112. Caropreso MF, Matwin S. Beyond the Bag of Words: A Text Representation for Sentence Selection. In: Lamontagne L, Marchand M, editors. *Advances in Artificial Intelligence*, Berlin, Heidelberg: Springer; 2006. DOI: 10.1007/11766247_28.

113. Juluru K, Shih HH, Keshava Murthy KN, Elnajjar P. Bag-of-Words Technique in Natural Language Processing: A Primer for Radiologists. *RadioGraphics* 2021; **41**(5): 1420–1426. DOI: 10.1148/rg.2021210025.

114. HaCohen-Kerner Y, Miller D, Yigal Y. The influence of preprocessing on text classification using a bag-of-words representation. *PLOS ONE* 2020; **15**(5): e0232525. DOI: 10.1371/journal.pone.0232525.

115. Forster KI. Left-to-right processes in the construction of sentences. *Journal of Verbal Learning and Verbal Behavior* 1966; **5**(3): 285–291. DOI: 10.1016/S0022-5371(66)80032-4.

116. Davies M. Corpus of Contemporary American English (COCA) 2017. DOI: 10.7910/DVN/AMUDUW.

117. Segbers J, Schroeder S. How many words do children know? A corpus-based estimation of children's total vocabulary size. *Language Testing* 2017; **34**(3): 297–320. DOI: 10.1177/0265532216641152.

118. Beer FP, Johnston E. R. *Scienza delle costruzioni. Introduzione alla meccanica dei materiali*. McGraw-Hill; 1997.

119. Zienkiewicz OC, Taylor RL, Zhu JZ. *The Finite Element Method: Its Basis and Fundamentals*.
120. Dassault Systèmes. *ABAQUS 6.14*. Vélizy-Villacoublay, Île-de-France, France: 2014.

Figures

Tables

Annexes

A. Development of the NLP methodology

A.1 Complete set of 50 training sentences from the COCA database

S. #	Sentence
S1	will you be my friend
S2	you and I will always be friends
S3	today is the first of november
S4	I saw a bear today
S5	she is in her room
S6	let go to the park
S7	I have a few questions
S8	I like her too
S9	it is sunny outside
S10	I really like it here
S11	that door is open
S12	this letter is for you
S13	you are really nice
S14	he is my brother
S15	I want to go with you
S16	I watch movies on my ipad
S17	what will you do now
S18	can I say something
S19	this is my favourite cookie
S20	can you pick me up at the mall
S21	I am sorry but she is away
S22	we are going to watch a movie
S23	this is his box
S24	this card came from my cousin
S25	that is a really cool trick

S. #	Sentence
S26	that is not what I want
S27	I cannot open it
S28	will you come by and see me
S29	she is very happy
S30	do you like blue or yellow
S31	her role as an english teacher is very important
S32	what are you thinking of
S33	I want to go there
S34	this is their house
S35	what can I do for you
S36	can you get me my eyeglasses
S37	what if I fail
S38	would you help me out
S39	I have her book
S40	all my favourite books are on this shelf
S41	my mom is coming to visit
S42	what is this movie about
S43	do you know where this place is
S44	I will help you find that place
S45	I live up in the mountains
S46	she is one of my english teachers
S47	there was a time I liked to play golf
S48	there are so many things I want to learn
S49	this is the year I am going to learn english
S50	I am so sorry

A.2 Set-up parameters of the GA used in the K-means clustering

The following parameters were set in the user interfaces 1 and 2 of the implemented GA (cf. Fig. 56). The settings of the modified roulette method in the user interface 3 were set as in Fig. 56.

	Interface 1			Interface 2			
K	r_{min}	r_{max}	g	G_{max}	N	p_c [%]	p_m [%]
2	0	1	4	200	100	80	40
3	0	1	4	400	100	80	40
4	0	1	4	600	100	80	40
5	0	1	4	800	100	80	40
6	0	1	4	1000	100	80	40
7	0	1	4	1200	100	80	40
8	0	1	4	1400	100	80	40

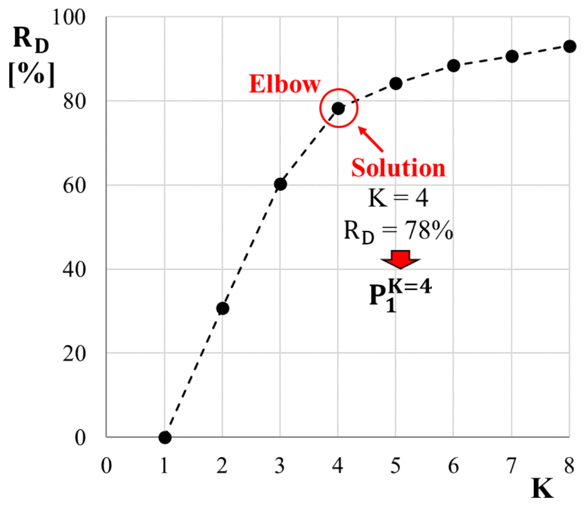
A.3 Set-up parameters of the ANN

The following parameters were set in the user interfaces 1 and 2 of the implemented ANN (cf. Fig. 39). The symbols η and μ respectively indicate the learning rate and the momentum (cf. Table 34).

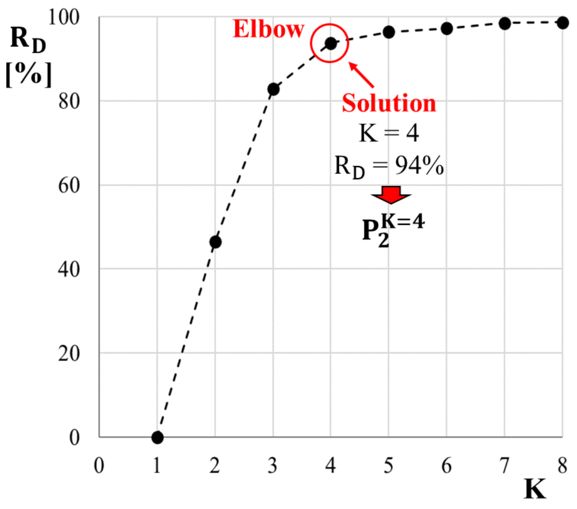
	Interface 1					Interface 2		
Test #	Number of hidden neurons	η hidden layer	η output layer	μ hidden layer	μ output layer	Calibration/ validation division [%]	Maximum number of epochs	Epoch interval of mixing
1	5	0.2	0.15	0.3	0.3	75	200	1
2	10	0.2	0.15	0.3	0.3	75	250	1
3	15	0.2	0.15	0.3	0.3	75	300	1
4	20	0.2	0.15	0.3	0.3	75	350	1
5	25	0.2	0.15	0.3	0.3	75	400	1
6	30	0.2	0.15	0.3	0.3	75	450	1

A.4 Pre-training of the improved NLP algorithm: curves K-R_D and resulting partitioning levels inherent to CFs^1 and CFs^2

CFs^1 = {“I”, “is”, “you”}



CFs^2 = {“my”, “this”, “to”}



$P_1^{K=4}$			ASs’ features (CFs^1)		
Cluster	**Training samples (sentences) #**	**AS**	**I**	**is**	**you**
C1.1	1, 3, 4, 5, 6, 7, 8, 10, 13, 16, 19, 20, 22, 24, 25, 27, 28, 30, 33, 36, 38, 39, 40, 42, 45, 46, 50	AS1.1	0	0.07	0.07
C1.2	9, 11, 14, 21, 23, 29, 31, 34, 41, 43	AS1.2	0	0.67	0.07
C1.3	12, 15, 17, 32, 35, 44	AS1.3	0.13	0.13	0.73
C1.4	2, 18, 26, 37, 47, 48, 49	AS1.4	0.53	0.07	0

$P_2^{K=4}$			ASs’ features (CFs^2)		
Cluster	**Training samples (sentences) #**	**AS**	**my**	**this**	**to**
C2.1	1, 14, 16, 19, 24, 36, 46	AS2.1	0.67	0	0
C2.2	2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 17, 18, 20, 21, 23, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 37, 38, 39, 44, 45, 50	AS2.2	0	0.07	0
C2.3	6, 15, 22, 33, 41, 47, 48, 49	AS2.3	0	0	0.67
C2.4	40, 42, 43	AS2.4	0.07	0.67	0

A.5 Fine-training of the improved NLP algorithm: results of the calibration tests inherent to $\mathbf{P_1^{K=4}}$ and $\mathbf{P_2^{K=4}}$ and choice of the ANNs

The results of the calibration tests are reported below in terms of E_{ep}^{v} and ΔE_{ep}^{v}, in the same way as in Table 77 (cf. section 3.1.2.2).

<table>
<tr><th colspan="4">$\mathbf{P_1^{K=4}}$</th></tr>
<tr><th>Test #</th><th>Number of hidden neurons</th><th>$\mathbf{E_{ep}^{v}}$</th><th>$\mathbf{\Delta E_{ep}^{v}}$ [%]</th></tr>
<tr><td>1</td><td>5</td><td>0.1</td><td>---</td></tr>
<tr><td>2</td><td>10</td><td>0.09</td><td>-10</td></tr>
<tr><td>3</td><td>15</td><td>0.08</td><td>-11</td></tr>
<tr><td>4</td><td>20</td><td>0.08</td><td>0</td></tr>
<tr><td>5</td><td>25</td><td>0.08</td><td>0</td></tr>
<tr><td>6</td><td>30</td><td>0.08</td><td>0</td></tr>
</table>

← **Adopted for the NLP algorithm**
Accuracy ≈ 92%

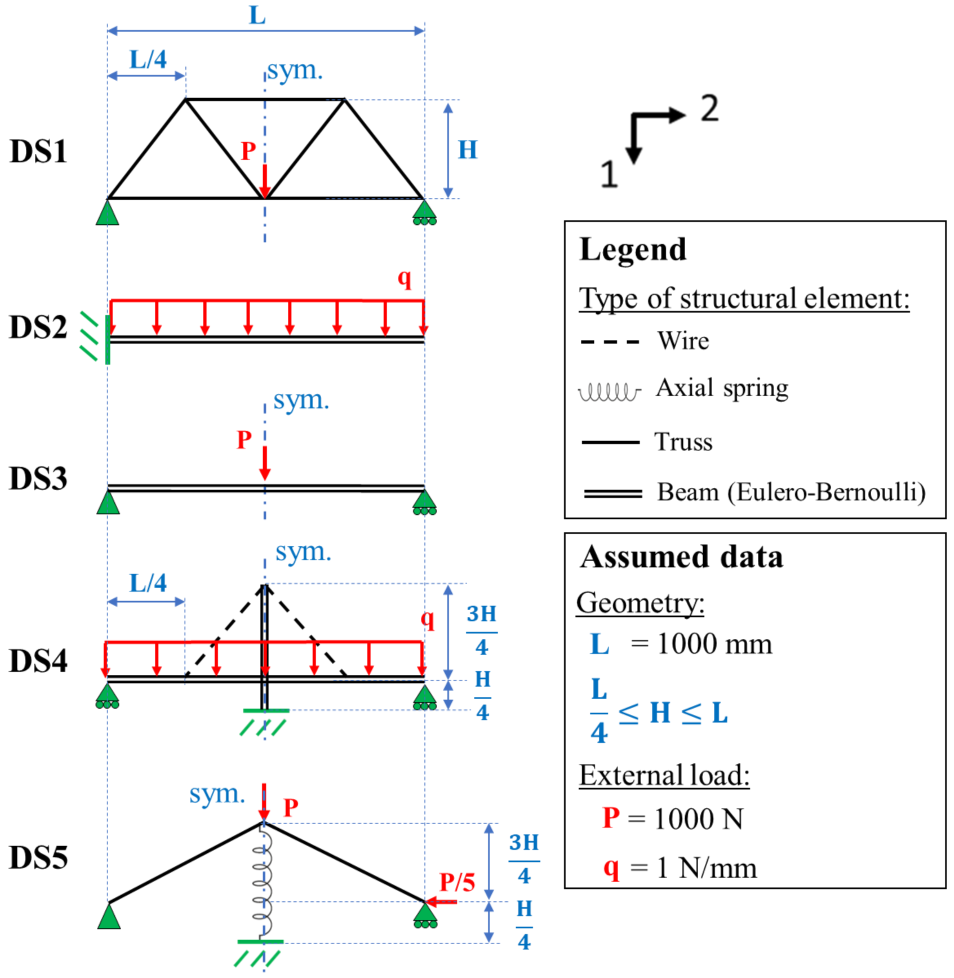
<table>
<tr><th colspan="4">$\mathbf{P_2^{K=4}}$</th></tr>
<tr><th>Test #</th><th>Number of hidden neurons</th><th>$\mathbf{E_{ep}^{v}}$</th><th>$\mathbf{\Delta E_{ep}^{v}}$ [%]</th></tr>
<tr><td>1</td><td>5</td><td>0.07</td><td>---</td></tr>
<tr><td>2</td><td>10</td><td>0.07</td><td>0</td></tr>
<tr><td>3</td><td>15</td><td>0.07</td><td>0</td></tr>
<tr><td>4</td><td>20</td><td>0.06</td><td>-14</td></tr>
<tr><td>5</td><td>25</td><td>0.06</td><td>0</td></tr>
<tr><td>6</td><td>30</td><td>0.06</td><td>0</td></tr>
</table>

← **Adopted for the NLP algorithm**
Accuracy ≈ 94%

B. Development of the demonstrator

B.1 FEM models used for the verification of the SPs

The implemented FEM models are illustrated below and are based on the five DSs reported in Fig. 78. The assumed values of L, P, q (see figure) and those of E_i, rho_i, A_i and Jb_i (see table below) enable to obtain numerical results which are coherent with the assumptions made in section 3.2.1.2 (cf. Table 83). Varying H enables to test different configurations. For the table below check the use limitations concerning element types in Table 86.



	Types of structural element used in the FEM models above			
Characteristic	**Beam**	**Truss**	**Wire**	**Spring**
Size of the finite element	10 mm	The finite element corresponds to the structural element (see figure above)		
Type of the finite element in ABAQUS	B21	T2D2		SpringA
Elastic modulus (E)	200,000 MPa (steel)			20 N/mm
Density (rho)	$7.8 \cdot 10^{-9}$ t/mm^3 (steel)			---
Section area (A)	10 mm^2		1 mm^2	---
Bending inertia (Jb)	10,000 mm^4	---		

B.2 Examples of SP verification via the FEM models

An example for each SP type is provided (cf. Table 89). U1 and U2 represent the absolute values of the maximum displacements along the directions 1 and 2, respectively (cf. Table 85).

Example 1: SP of type 1 (general increase, cf. Table 89). Type 2 (general decrease) is analogous.

DS name	Spec.	SP to verify		SP interpretation
		Part 1 (action)	Part 2 (ind.)	
DS1	U1_-	A_+		Increase A
	U2_-			

Lacking indications (part 2), the section area A is increased in all truss elements of DS1 (cf. section 3.2.2.3). The results in the table below are obtained with the FEM model of DS1 (cf. annex B.1), by studying four different values of H. The section area A (cf. annex B.1) was arbitrarily increased by 20%.

	A = 10 mm^2		1.2 A = 12 mm^2	
H	U1 [mm]	U2 [mm]	U1 [mm]	U2 [mm]
L/4	0.73	0.25	0.61 (< 0.73)	0.21 (< 0.25)
L/2	0.44	0.13	0.37 (< 0.44)	0.1 (< 0.13)
3L/4	0.48	0.08	0.4 (< 0.48)	0.07 (< 0.08)
L	0.57	0.06	0.48 (< 0.57)	0.05 (< 0.06)

An increase of A produces a decrease of both U1 and U2 ➔ **the SP is verified for both U1_- and U2_-**

Example 2: SP of type 3 (local increase, cf. Table 89). Type 4 (local decrease) is analogous.

DS name	Spec.	SP to verify		SP interpretation
		Part 1 (action)	Part 2 (ind.)	
DS5	U1_-	E_+	spring_w	Increase E of the spring
	U2_-			
	Fn_+			

The results in the table below are obtained with the FEM model of DS5 (cf. annex B.1), by studying four different values of H. The stiffness of the spring (cf. annex B.1) was arbitrarily increased by 20%.

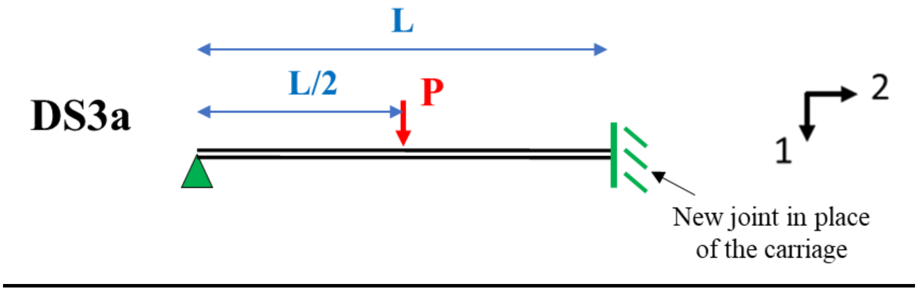
	E spring = 20 N/mm			1.2 E spring = 24 N/mm		
H	U1 [mm]	U2 [mm]	Fn [Hz]	U1 [mm]	U2 [mm]	Fn [Hz]
L/4	42.5	31.8	107	35.4 (< 42.5)	26.4 (< 31.8)	117 (> 107)
L/2	35	52.3	68.4	29.2 (< 35)	43.6 (< 52.3)	74.9 (> 68.4)
3L/4	27.5	61.5	45.7	22.9 (< 27.5)	51.2 (< 61.5)	50 (> 45.7)
L	20	59.4	32.5	16.7 (< 20)	49.4 (< 59.4)	35.6 (> 32.5)

An increase in E of the spring produces a decrease of U1 and U2, and an increase in Fn ➔ **the SP is verified U1_-, U2_- and Fn_+**

Example 3: SP of type 5 (replacement, cf. Table 89).

DS name	Spec.	SP to verify		SP interpretation
		Part 1 (action)	Part 2 (ind.)	
DS3	U1_-	joint_+	carriage_-	Add joint in place of carriage
	Fn_+			

The following FEM model, named DS3a, is obtained from DS3 (cf. annex B.1) by replacing the carriage with a joint, and it is used to verify the SP.



The results below are obtained with the FEM models DS3 (cf. annex B.1) and DS3a

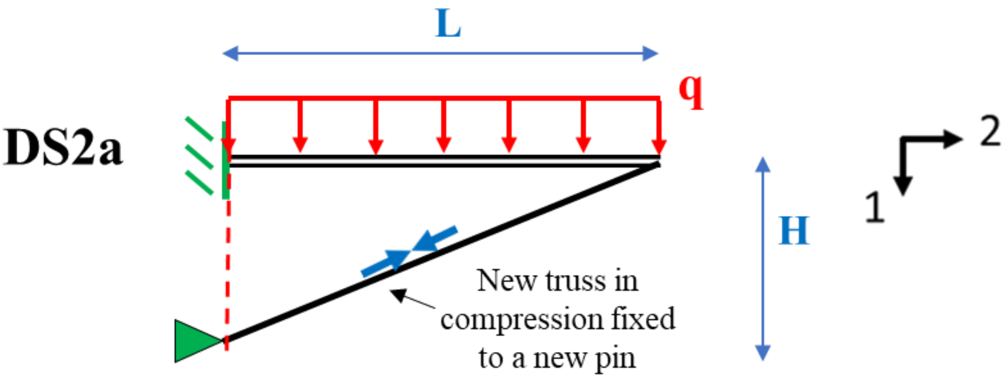
DS3		DS3a	
U1 [mm]	Fn [Hz]	U1 [mm]	Fn [Hz]
10.4	251.5	4.7 (< 10.4)	392.9 (> 251.5)

The replacement of the carriage with a joint produces a decrease of U1 and an increase of Fn ➔ **1)** the SP is verified for both U1_- and Fn_+ **2)** The model DS3a will be used as supporting DC for the verified SP

Example 4: SP of type 6 (structural addition, cf. Table 89).

DS name	Spec.	SP to verify		SP interpretation (cf. Table 89)
		Part 1 (action)	Part 2 (ind.)	
DS2	U1_-	truss_compression_+	pin_+	Add truss in compression fixed to a new pin
	Fn_+			

The following FEM model, named DS2a, is constructed based on DS2 (cf. annex B.1), and it is used to verify the SP. It is assumed to add the new pin on the left side, in alignment with the beam joint. The properties of the added truss in terms of A and E are the same as the beam (cf. annex B.1.



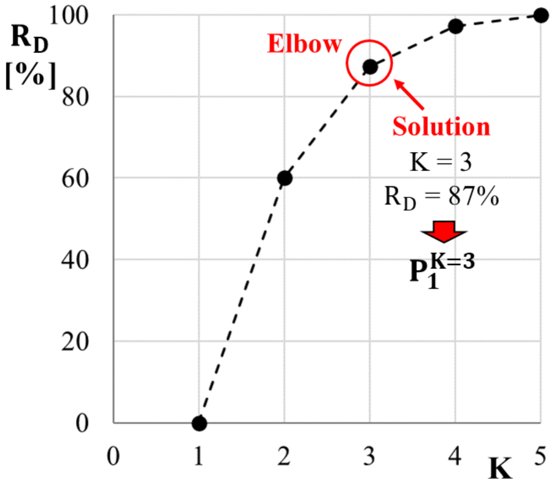
The results below are obtained with the FEM models DS2 (cf. annex B.1) and DS2a, studying for different values of H.

	DS2		DS2a	
H	U1 [mm]	Fn [Hz]	U1 [mm]	Fn [Hz]
L/4	62.5	89.6	5.9 (< 62.5)	184.7 (> 89.6)
L/2			3.5 (< 62.5)	294.8 (> 89.6)
3L/4			3.1 (< 62.5)	340.8 (> 89.6)
L			3 (< 62.5)	356.7 (> 89.6)

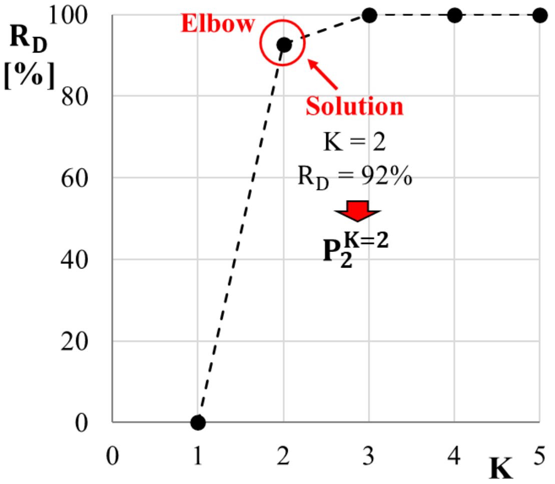
The added truss produces a decrease of U1 and an increase of Fn ➔ **1)** the SP is verified for both U1_- and Fn_+ **2)** The model DS2a will be used as supporting DC for the verified SP

B.3 Pre-training: curves K-R$_D$

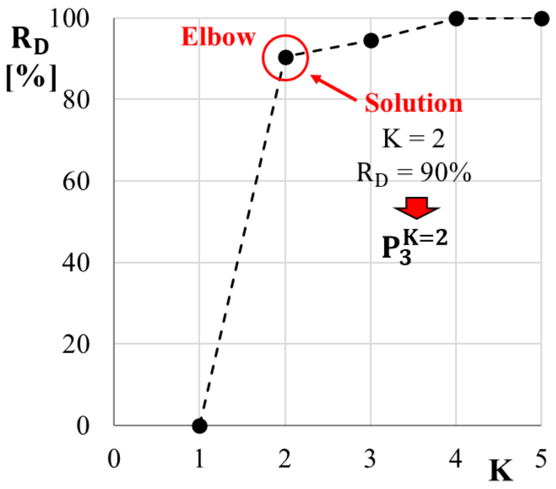
CFs^1 = {free, linear, close}



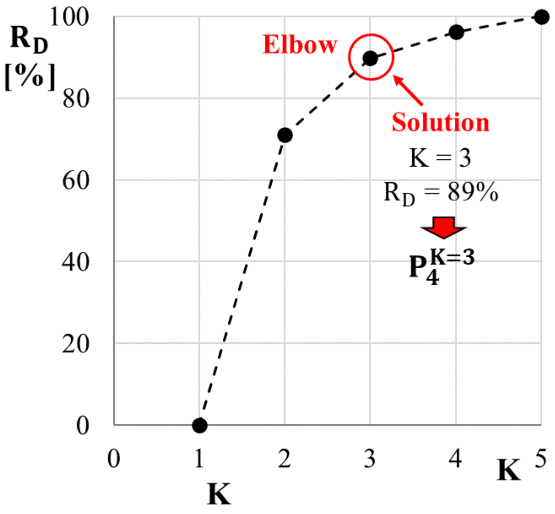
CFs^2 = {straight, extremity}



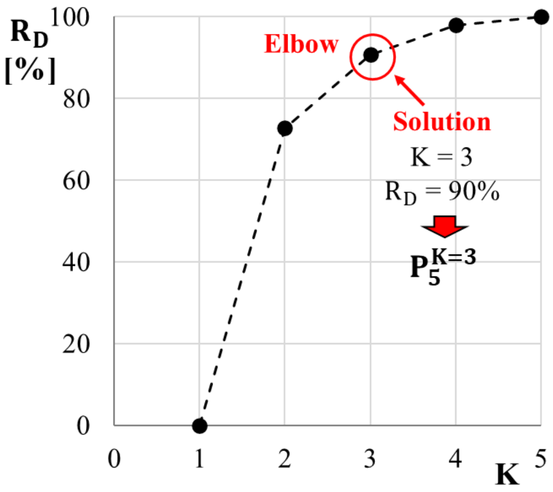
CFs^3 = {wire, spring, truss, beam}



CFs^4 = {carriage, pin, joint}



CFs^5 = {traction, compression, bending}



B.4 Fine-training: results of the calibration tests inherent to the five partitioning levels (cf. annex B.3) and choice of the ANNs

The results of the calibration tests are reported below in terms of E_{ep}^{v} and ΔE_{ep}^{v}, in the same way as in annex A.5 for the improved NLP methodology.

$\mathbf{P_1^{K=3}}$			
Test #	**Number of hidden neurons**	$\mathbf{E_{ep}^{v}}$	$\mathbf{\Delta E_{ep}^{v}}$ **[%]**
1	5	0.13	---
2	10	0.09	-31
3	15	0.09	0
4	20	0.09	0
5	25	0.09	0
6	30	0.09	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 91%

$\mathbf{P_2^{K=2}}$			
Test #	**Number of hidden neurons**	$\mathbf{E_{ep}^{v}}$	$\mathbf{\Delta E_{ep}^{v}}$ **[%]**
1	5	0.07	---
2	10	0.07	0
3	15	0.07	0
4	20	0.07	0
5	25	0.07	0
6	30	0.07	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 93%

$\mathbf{P_3^{K=2}}$			
Test #	**Number of hidden neurons**	$\mathbf{E_{ep}^{v}}$	$\mathbf{\Delta E_{ep}^{v}}$ **[%]**
1	5	0.15	---
2	10	0.09	-40
3	15	0.08	-11
4	20	0.08	0
5	25	0.08	0
6	30	0.08	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 92%

$\mathbf{P_4^{K=3}}$			
Test #	**Number of hidden neurons**	$\mathbf{E_{ep}^{v}}$	$\mathbf{\Delta E_{ep}^{v}}$ **[%]**
1	5	0.12	---
2	10	0.10	-16
3	15	0.10	0
4	20	0.10	0
5	25	0.10	0
6	30	0.10	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 90%

$\mathbf{P_5^{K=3}}$

Test #	Number of hidden neurons	$\mathbf{E_{ep}^{v}}$	$\mathbf{\Delta E_{ep}^{v}}$ [%]
1	5	0.11	---
2	10	0.10	-9
3	15	0.09	-10
4	20	0.09	0
5	25	0.09	0
6	30	0.09	0

← **Adopted for the NLP algorithm**
Accuracy ≈ 91%