Data Driven Approach to Enhancing Efficiency and Value in Healthcare

Richard E. Guerrero Ludueña

Part I

Background information

Chapter 2

Background information

This chapter presents a review of the three main methods we will use in this thesis: Metaheuristic Optimisation, System Modelling and Simulation, and Data Analytics. Main focuses of this review is to present a introduction to the methods, a summary of different approaches described in the literature, and applications to healthcare.

The review is organised as follows. Metaheuristic Optimisation is discussed in Section 2.1; Machine Learning is introduced in Section 2.2; Modelling and Simulation is presented in Section 2.3; and a review of Data Analytics is presented in Section 2.4; Section 2.5 presents the conclusions of this review; Finally, Section 2.6 summarises this chapter.

2.1 Metaheuristic Optimisation

This section presents a review of metaheuristics algorithms and its application to solve optimisation problems in healthcare. The main focus of this section is a discussion of Genetic Algorithms, a metaheuristic algorithm applied in the Chapter 5. A review of the general principles of optimisation and metaheuristic algorithms, similarities and differences with other academic fields, and an introduction to alternative metaheuristic algorithms are also presented.

2.1.1 Optimisation

One of the fundamental challenges in modern organisations is the efficient utilisation of expensive resources. Optimisation is one of the most popular quantitative decision methodologies, since we can find optimisation problems everywhere, from workforce planning to surgery scheduling in a hospital [128, 138]; from a modern controller design in an industrial evaporation system [166] to choosing the optimum location and number of ambulances [156]. In these examples, the aim is to achieve an objective or to optimise something such as profit, time, energy or travels. As resources are always limited in real-world applications, we have to find solutions to optimally use these valuable resources under various constraints. Mathematical optimisation, mathematical programming or simply optimisation is the selection of the best element (with regard to some criteria) from a set of available alternatives.

Every process has a potential to be optimised, examples of real-life optimisation are minimisation of travel time, process costs, and risk or maximisation of profit, quality, and efficiency. A large number of real-life optimisation problems are complex and difficult to solve; they cannot be solved in an exact manner within a reasonable amount of time. Using approximate algorithms is the main alternative to solve this class of problems [204].

A mathematical program is an optimisation problem of the form:

$$\begin{aligned} &maximize && f_i(x), && (i = 1, 2, ..., I) \\ &subject \quad to && g_j(x) \leq 0, && (j = 1, 2, ..., J) \\ & && h_k(x) = 0, && (k = 1, 2, ..., K) \end{aligned} \tag{2.1}$$

where $f_i(x)$, $g_j(x)$, and $h_k(x)$ and

$$x \in X \tag{2.2}$$

Here X is a subset of $\mathbb{R}^n$ and is in the domain of f_i, g_j and h_k, which map into real spaces. The functions $f_i(x)$ are called *objective function*. The components x of X are called *decision variables*, and they can be real continuous, discrete or the mixed of these two. The space extend over the decision variables is called *design space*, while the space formed by the objective functions values is called *solution space*. The *solution space* S, also named *feasible region*, *feasible set* or *search space* is the *set* of all *possible* points (*sets* of values of the choice variables) of an optimisation problem that satisfy the problem's constraints.

The relations, $x \in X$, $g_j \leq 0$, and $h_k = 0$ are called *constraints*, and the functions f_i where $i = 1, 2, ..., I$ are called *objective function.*

A point x is *feasible* if it is in X and satisfies the constraints $g_j(x) \leq 0$, and $h_k(x) = 0$. A point x^* is *optimal* if it is feasible and if the value of the objective function is not less that of any other feasible solution: $f(x^*) \geq f(x)$ for all feasible x. The *sense of optimisation* is presented here as *maximisation*, but it could just as well be *minimisation*, with the appropriate change in the meaning of optimal solution: $f(x^*) \leq f(x)$ for all feasible x.

An optimisation problem can be classified according to the number of objectives as single objective (in the case of $M = 1$) or multionjective ($M > 1$). A multiobjective optimisation is also referred to multi-attributes or multicriteria. In real-world problems, most optimisations tasks are multiobjective [129]. Another classification of optimisation is in terms of number of constrains $J + K$. If there is no constraint ($J = K = 0$), the problem is called an unconstrained optimisation problem [195]. If $J = 0$ and $K \geq 1$, the problem is called equality-constrained problem [197, 126]; while $K = 0$ and $J \geq 1$, represent a problem called inequality-constrained optimisation problem [181, 68].

Optimisation problems can also be classified as linear or nonlinear based on the actual function forms. In a linearly constrained problem, the constrains g_j and h_k are all linear. If both the constraints and the objective functions are linear, it becomes a linear programming problem. If all f_i, g_j and h_k are nonlinear, the problem is called nonlinear optimisation problem [232, 119].

The size is an alternative approach to classify optimisation problems, measured in terms of the number of unknown variables or the number of constraints: *small-scale* are a class of problems with about five or fewer unknowns variables and constraints; *intermediate-scale* are problems with about five to a hundred or a thousand variables; and a *large-scale problems* are problems with thousands or even millions of variables and constraints [198].

In optimisation, a mathematical program is often extended to indicate a *family* of mathematical programs over a *parameter space*, say P. This involves extending the domain of the functions, f, g and h, and a semi-colon is used to separate the decision variables from the parameters.

$$\begin{aligned} max \qquad & f(x,p) \\ & g(x,p) \leq 0 \\ & h(x,p) = 0 \\ & x \in X \end{aligned} \tag{2.3}$$

2.1.1.1 Combinatorial Optimisation (CO)

Many real-world optimisation problems consist of the search for a best configuration of a set of variables to achieve goals. As noted, optimisation problems can be divided based on the type or existence of constraints (e.g. constrained and unconstrained optimisation problems), number of objective functions (e.g. single-objective and multi-objective programming), nature of equations (linear programming, nonlinear programming, geometric programming, quadratic programming, etc.), values of decision variables (e.g. real-valued variables, discrete variables).

If we focus on the classification of optimisation problems based on the value of the decision variables, Combinatorial Optimisation (CO) is defined as a type of problem where we are looking for an object from a finite - or possibly countably infinite - set of variables [105, 121, 32]. This objective is typically an integer number, a subset, a permutation, or a graph structure [18]. Examples for CO problems are the Travelling Salesman problem (TSP), the Quadratic Assignment Problem (QAP), Timetabling and Scheduling problems [18].

2.1.2 Different approaches for solving optimisation problems

Once an optimisation problem has been formulated correctly, the following step is to find a solution by using and exact method, approximate algorithms, or heuristic approach, as appropriate.

Since the solution space S is finite, any optimisation problem could be exactly solved by an algorithm that simply enumerates all elements in S and outputs one among those elements corresponding to the best objective function value x^*. Unfortunately, since the number of feasible solutions $\mid S \mid$ usually grows exponentially with the size of the instance to be solved, such a naive approach is not efficient and surely not applicable to solve real-world applications of practical interest. Several optimisation problems can be solved in polynomial time (problems in P), but many of them are hard or computationally intractable ($NP-complete$ problems), or *non-deterministic polynomial time*. Although any given solution to an $NP-complete$

problem can be verified quickly (in polynomial time), there is no known efficient way to locate a solution in the first place and the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a consequence, determining whether or not it is possible to solve these problems quick, called the P versus NP problem, is one of the principal unsolved problems in computer science today [43, 70].

Classical approaches to solve an optimisation problem are Branch & Bound [125], and Dynamic Programming [15]. They are divide-and-conquer methods, since both solve a problem by combining the solutions to its subproblems.

If $P \neq NP$, we cannot simultaneously have algorithms that (1) find optimal solutions (2) in polynomial time (3) for any instance [224]. At least one of the these requirements must be relaxed in any approach to dealing with and NP-complete optimisation problem. Relaxing the "for any instance" requirement, and finding polynomial-time algorithms for special cases of the problem at hand is one of the approaches. The second approach is to relax the requirement of polynomial-time solvability, aiming to find optimal solutions to problems by clever exploration of the full set of possible solutions to a problem, accepting that the algorithm will terminate in *any* reasonable amount of time. A more common approach identified by Williamson is to relax the requirement of finding an optimal solution, and instead settle for a solution that is "good enough", especially if it can be found in seconds or less. By relaxing the requirement of finding an optimal solution, NP-complete problems are often addressed using *approximation algorithms* and *heuristic and metaheuristics* methods.

By using an approximation algorithm the aims is to find a suboptimal solution that closely approximates the optimal solution in terms of its value, providing an approximation-guarantee α on the quality of the solution found.

2.1.3 Metaheuristic optimisation

As discussed, many optimisation problems of practical as well as theoretical importance consist of the search for a best configuration of a set of variables to achieve goals [18]. Algorithms developed to tackle optimisation problems are classified as *exact* (also called completed methods) or *approximate* algorithms [74]. Approximate algorithms are not able to certify the optimality of the solutions they find; exact procedures theoretically can guarantee to find for every finite size instance of a problem an optimal solution in a limited time [162]. For problems that are $NP-hard$ [70], no polynomial time algorithm exists, assuming that $P \neq NP$. Therefore, complete methods might need exponential computation time in the worst-case. In approximate methods we sacrifice the guarantee of finding good solutions in a significantly reduced amount of time [18].

Approximate algorithms can be classified as specific heuristics and metaheuristics [204]. An heuristic is defined as an approximate solution technique designed and applicable to a particular problem [73]. The term heuristic is sometimes used to refer to a whole search algorithm and is sometimes used to refer to a particular decision process sitting with some repetitive control structure [28]. Metaheuristic was defined for the first time in 1986 [76], and derives from the composition of two Greek words. *Heuristic* derives from the verb *heuriskein* which means "to find", while the suffix *meta* means "beyond, in an upper level" [18]. In [201], the history of metaheuristics is described based on five distinct periods:

- Pre-theoretical period (until c. 1940), heuristics and even metaheuristics are used but not formally studied.
- Early period (c. 1940 - c. 1980), formal studies on metaheuristic appear.
- Method-centric period (c. 1980 - c. 2000), many different methods proposed.
- Framework-centric period (c. 2000 - present), metaheuristics are more usefully described as frameworks, and not as methods.
- Scientific period (the future), metaheuristics becomes a science.

Due to the computational complexity of hard optimisation problems, especially in the presence of large scale problem instances, there has been a sizeable study of various types of heuristics and metaheuristics such as simulated annealing [110, 56], tabu search [76, 78], evolutionary algorithms like genetic algorithms [79], colony optimisation [51], and scatter search [77], to name a few.

Metaheuristics are defined as a class of methods commonly applied to suboptimally solve computationally intractable optimisation problems. Metaheuristic has been also defined as an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions [159]. A metaheuristic may combine different concepts for exploring the search space and uses learning strategies to structure the information in order to find efficiently near-optimal solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.

Four main components of metaheuristics are: initial space of Solutions; search engines; learning and guideline strategies; and finally management of information structures [159].

Metaheuristics has been located between Artificial Intelligence (AI) and Operational Research (OR) [28], as part of machine learning and soft computing [211], and also as a branch of optimisation in Computer Science (CS) and applied mathematics that are related to algorithms and computational complexity theory [204].

Metaheuristic has been defined in numerous ways by different authors. In this thesis, we adopt the definition presented in [77, 74].

Definition 2.1.1. Metaheuristics are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space.

2.1.4 Genetic Algorithms

Figure 2.1 shows a classification of optimisation approaches based on the technique used to search for solutions.

Guided random search techniques are method based on random searches in locating the optimal solution. These methods are different from pure ”random walk” methods in the sense that they use information from the previous iteration in locating the next best point. These methods are hence classified under guided random search methods. The *guided random search techniques* can be subclassified into Evolution-inspired methods, Other Nature-inspired methods, and Logical Search techniques. Genetic Algorithms (GA), Evolutionary Programming, and Evolution Strategies are three types of Evolutionary Algorithms. Simulated Annealing (SA) and Ant Colony Optimisation (ACO) are two examples of other Natural based techniques. Both GA and ACO techniques use population of points in the search space and hence have a better change of locating the global optima solution. The GA technique mimics the biological process (genetics) whereas the ACO technique is based on the idea of natural phenomena of the ants.

Genetic Algorithm (GA) was introduced by Holland in 1975 [94]. GA are self-adapting strategies for searching, based on the random exploration of the solution space completed with a memory component which enables the algorithms to learn

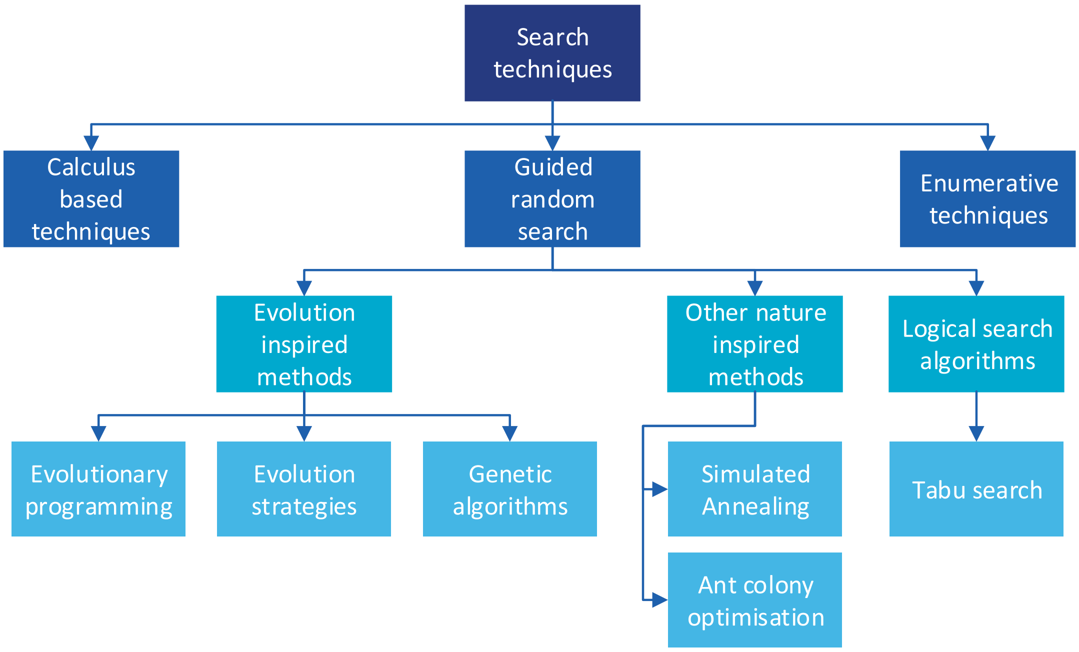

Figure 2.1: Family three of Optimisation approaches based on search technique.

the optimal search path from experience [202]. The algorithm is based on the natural selection process proposed by Darwin, whereby organisms evolve by rearranging genetic material to survive in environments confronting them ('Survival of the fittest'). GA provide optimal or near optimal solutions for both constrained and unconstrained optimisation problems, repeatedly modifying a population of individual solutions. GA is part of the Evolutionary Algorithms, a classic example of heuristic search algorithms.

GA can be divided into four main parts: definition of an appropriate structure to represent the solution; determination of the fitness function; design of genetic operators; and determination of probabilities controlling the genetic operators. A classic GA can be described as follow:

1. Start with a randomly generated population of n l-bit chromosomes (candidates solutions to a problem).

2. Calculate the fitness $f(x)$ of each chromosome x in the population.

3. Repeat the following steps until n offspring have been created:

(a) Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done *with replacement*, meaning that the same chromosome can be selected more than once to become a parent.

(b) With probability p_c (the *crossover probability* or *crossover rate*), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.

(c) Mutate the two offspring at each locus with probability p_m (the *mutation probability* or *mutation rate*), and place the resulting chromosomes in the new population.

4. Replace the current population with the new population.

5. Go to step 2.

Each iteration of this process is called *generation*. The entire set of generations is called a *run*. At the end of a run there are often one or more highly fit chromosomes in the population. Since randomness plays a large role in each run, two runs with different random-number seeds will generally produce different detailed behaviours.

Procedure summarised at the Figure 5.2 describes the basis for most applications of GAs. There are a number of details to fill in, such as the size of the population and the probabilities of crossover and mutation, and the success of the algorithm often depends greatly on these details.

2.2 Machine Learning

Machine Learning (ML) is an interdisciplinary field that combines theories and techniques from computer science, optimisation and statistics, and is sub-field of Artificial Intelligence (AI) and a central part of Data Science (DS) [103, 194, 180].

In 1950, Turing asked, ”Can machines think?”, and since then, multiples definitions have been proposed [214]. The first know formal definition of ML was presented in 1959 [186]. The author presented ML as *field of study that gives computers the ability to learn without being explicitly programmed*. The popular definition *a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with the experience E* was proposed in [141]. ML has been also defined as

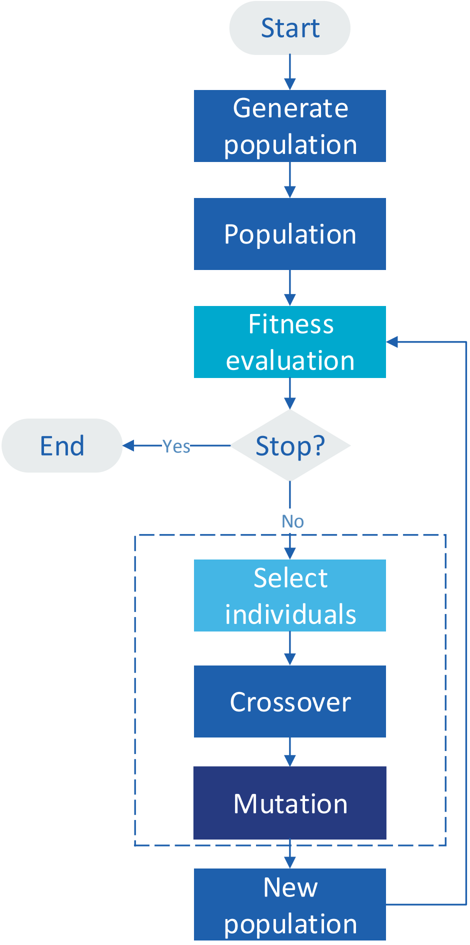

Figure 2.2: Flowchart for the Genetic Algorithm.

a discipline concerned with providing programs with the ability to learn and adapt [95]. The term *automated learning* was used as part of another definition of ML [194], where the author argues that in ML we wish to program computers so that they can *learn* from input available to them, and also, that the term ML refers to the automated detection of meaningful patterns in data. Jordan [103] proposes that ML addresses the question of how to build computers that improve automatically through experience, and extended the definition of ML to *a discipline dealing with the development of computer systems with the ability to automatically learn from experience; and with the identification of the fundamental laws (statistical, computational, information, theoretical) that govern all learning systems, including computers, humans, and organisations.* In [235], the authors suggested that *a ML problem is referred to as the problem of learning from experience with respect of some tasks and performance* *measure*, and said that ML depends on efficient learning techniques (algorithms), rich and/or large data, and powerful computing environments.

Based on the definitions presented above, in this thesis we define ML as:

Definition 2.2.1. Machine Learning is an interdisciplinary discipline that focuses on the implementation of computer software with capability to learn autonomously from inputs available to them, and to generalise the knowledge in order to adapt to new inputs.

The above definition (2.2.1) has five key elements:

1. *Interdisciplinary discipline.* ML combines theories and techniques from computer science, optimisation and statistics.

2. *Implementation of computer software.* ML is based on the realisation of an algorithms as a software through computer programming.

3. *Capability to learn autonomously.* ML aims with the implementation of algorithms to give computer software the ability to learn from experience without being explicitly programmed.

4. *From available inputs.* The ability to learn is directly related with the data provided to the ML implementation.

5. *Generalisation and adaptation.* Generalisation and adaptation are related with the ability of a ML implementation to learn from a sample of known data and applied knowledge to make predictions in a future output variable given new samples of unknown inputs.

In the recent years, both academics and companies have focused their attention on the relationship between ML and Data Analytics and Big Data (see for example [188, 99, 2, 235]). One of the main questions regarding ML is why not directly program a computer to solve a problem manually, using the traditional programming paradigm (see Figure 2.3). Two aspects have been identified as a key to decide when to use ML rather than directly programme a computer with every rule, command and response: problem complexity and the need for adaptivity [194].

Tasks performed by animals or humans, like driving, face or speech recognition, image understanding, and also tasks beyond human capabilities (for example, analysis of very large and complex data sets such as weather prediction; extracting medical knowledge from medical archives; analysis of 'omics' data, and web search engines),

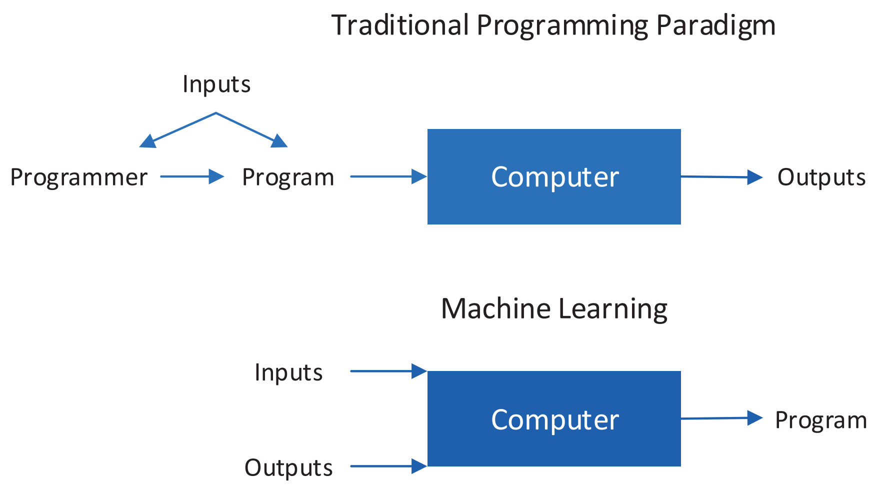

Figure 2.3: Machine Learning and the Traditional Programming Paradigm. Machine learning algorithms learn models from data, instead of formulating ’rules’ manually.

are two types of tasks too complex to be coded explicitly by humans. In all of those examples, ML techniques perform well once exposed to sufficient training examples [194].

The second aspect, *adaptability*, is related with the rigidity of programmed tools that stay unchanged once written down and installed. ML tools are programmes that adapt to their input data, and therefore are a solution to address tasks that change over time, or from one user to another [194].

Artificial Intelligence (AI) is defined as ”the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings” [39]. As noted, ML can be defined as part of AI. Both fields are related with the extraction of knowledge from experience and with the identification of meaningful patterns in complex data. These are characteristics that can be connected with human and animal intelligence [194]. Figure 2.4 presents the basic differences between AI and ML. ML focuses on knowledge extraction using data from previous experiences to complement human intelligence through the generalisation of learning. AI tries to build automated imitation of intelligent behaviour.

In [4], Data Mining (DM) is defined as the application of ML to large databases. In the past, an accepted DM definition was the *analysis* step as part of the *’Knowledge Discovery in Databases’* (KDD). There are different approaches to discovering

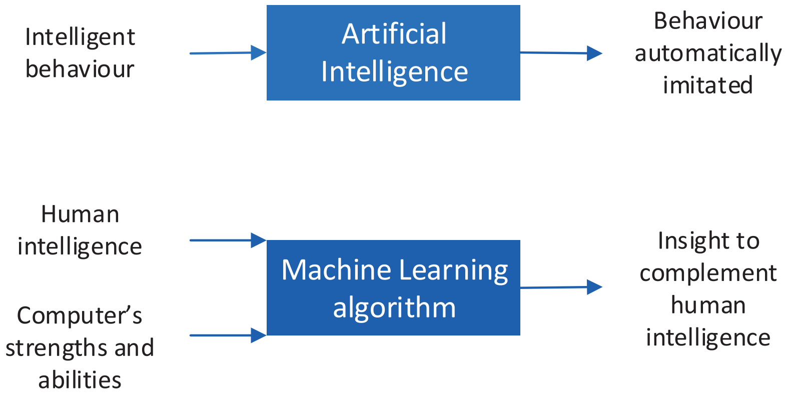
Intelligent behaviour
Artificial Intelligence
Behaviour automatically imitated
Human intelligence
Computer's strengths and abilities
Machine Learning algorithm
Insight to complement human intelligence

Figure 2.4: Machine Learning and Artificial Intelligence.

properties of data sets. Machine learning is one of them. Another one is simply looking at the data sets using visualisation techniques or Topological data analysis.

2.2.1 Machine Learning Process

ML application are summarised as data pre-processing, learning, and evaluation (see Figure 2.5) [235]. Before applying the ML models for the learning steps, in the data pre-processing phase, the data must be converted into a form that can be used as inputs to the learning step. At the learning phase, learning algorithms are choices and the model parameters are tunes, ML models are then applied to the pre-processed data to generate desired outputs. The final step of a ML application is the performance evaluation of the learned models. The evaluation results may lead to adjusting the parameters of chosen learning algorithms and/or selection different algorithms [235].

Figure 2.5 shows a framework for ML, where the component interacts with users, domain, and system. Users may interact with ML by providing domain knowledge, usability feedback, personal bias, and by leveraging learning outcomes to improve decision making; domain can serve both as a source of knowledge to guide ML and as the context of applying learning models; system architecture has impact on how learning algorithms should run and how efficient it is to run them, and simultaneously meeting ML needs may lead to a co-design of system architecture [235].

A cycle of four steps has been identified as part of a ML project [14]:

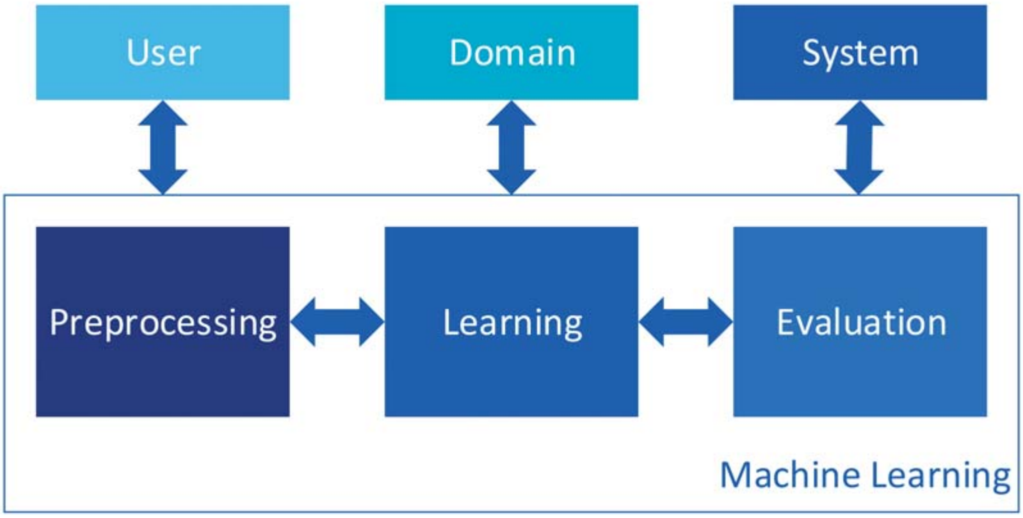

Figure 2.5: A framework of Machine Learning. Adapted from L. Zhou et al (2017) [235].

- *Acquisition.* Collate the data from different sources.
- *Prepare.* Data cleaning and checking for quality before any processing can take place.
- *Process.* Run machine tools (or machine learning routines).
- *Report.* Present the results.

2.2.2 A proposed new framework for Machine Learning applications

In this section, we propose a new process for ML applications. The definition of the section 2.2 (see Figure 2.5), assumes that the ML application begins with a dataset to train and evaluate the algorithms, and ends with the performance evaluation. However, a successful machine learning application requires additional steps. Figure 2.6 summarises the proposed framework for ML applications. The framework is based on five steps: *Definition, data preparation, learn, evaluation and application.*

1. Initiation step

    - *Understand domain.* Before beginning a ML project we need domain knowledge in order to ask the right question(s). Engagement with domain experts is an important part of this step.

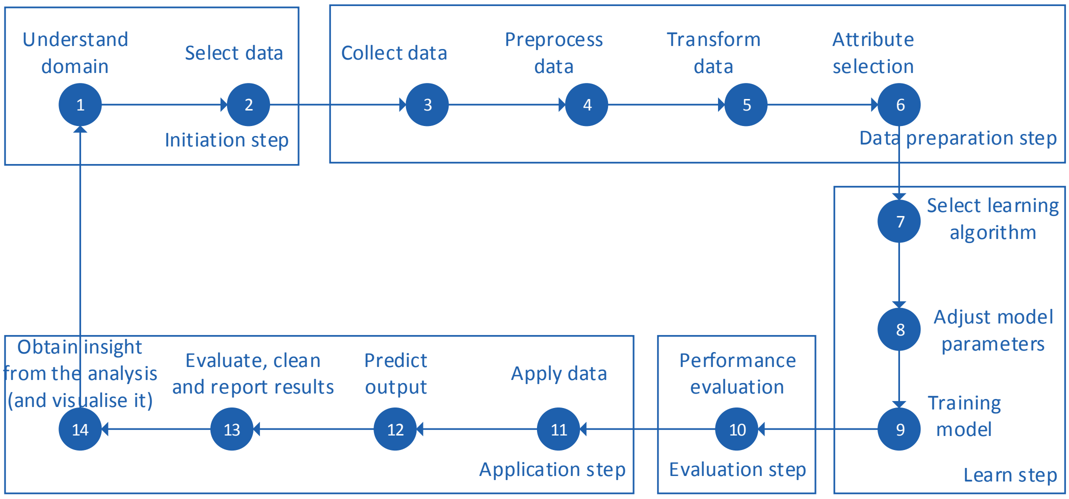

Figure 2.6: A proposed new framework for Machine Learning applications.

- *Question.* The most important step in a ML application (like any data-driven project) is the definition of the question, or a specific issue that needs investigation.
- *Select data.* Based on the domain knowledge, the next step is to define what information is essential for the analysis, what data sources are requested, and what data is available. Once the data sources have been identified, we need to exclude data unnecessary to address the question and/or domain experts support. The next step is to identify what data sources are essential for the analysis, and what data is available. At this stage, a record of criteria for data exclusion and assumptions is recommended to facilitate future analysis.

2. Data preparation step

- *Extract data.* The choice of methods for data extraction from the databases will depend on the variables to be measured, the source or sources to be used, and the resources available.
- *Pre-process data.* ML algorithms learn from data so it is critical that the right data is used to solve the problem. Even if the right data is available, it is important to make sure that it is in a useful scale, format and that even

meaningful features are included in the learning phase. Tabular data is the most common way of representing data in ML or Data Mining applications.

This whole procedure is the most time consuming and difficult process and is depicted in the Figure 2.7. Data pre-processing includes data munging and conversion into tabular data. Data Munging or Data Wrangle is a term used in Data Science, and is defined as the process of manual transformation of raw data into a usable format for more convenient consumption of the data with the help of semi-automated tools.

This phase is an important step in every ML (or Data Science) project, and sometimes is the most difficult and time consuming step. The level of difficulty depends on the quality of data (inconsistent data types, missing values, poor structure, outliers, etc.) and interaction with person delivering data (domain).

Data munging can often amount to 70% of overall project time & budget. In the practice, the first step of the data munging is get the data in a form that we can work with, plot the data to understand what is happening, and iterate between graphics and models to build a quantitative summary of the data. Some of the activities that comprise data munging are:

It is very likely that the selected ML tools will influence the data preprocessing to be performed.

- *Select features.* Deep knowledge of the problem domain is required to select which features or attributes to use to create a predictive model. Feature selection aims to reduce the number of attributes in the dataset, selecting a subset of features that are most relevant to the predictive model. There may be features that represent a complex concept that may be more useful to a machine learning method when split into different parts (decomposition), or there may be features that can be aggregated into a single feature that would be more meaningful to the problem we are trying to solve (aggregation).

  There are three general classes of feature selection algorithms: filter methods, wrapper methods, and embedded methods.

3. Learn step. Learning (a correct or possible incorrect) function or rule for a specific dataset is called inductive learning [180].

   - Select learning algorithm.

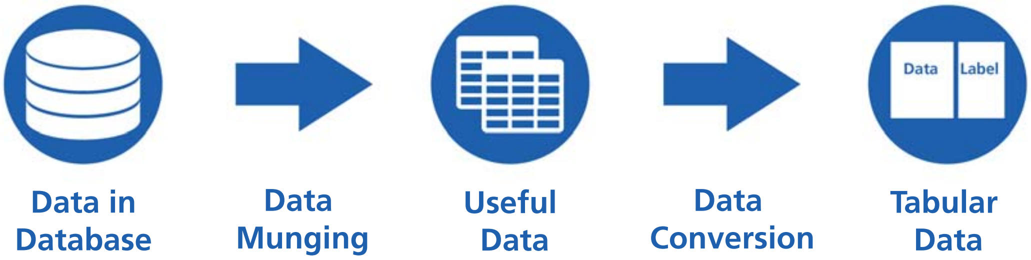

Figure 2.7: Data pre-processing in Machine Learning.

- Adjust model parameters.
- Training model. A machine learning model sis an abstraction of the question we are trying to answer or the outcome we want to predict.Models are trained and evaluated from existing data. When training a model, known data are used to adjust the model based on the data characteristics to get the most accurate answer.

4. Test step.
5. Evaluation step.
    - Evaluate performance. Once trained the model is evaluated using a randomly selected data set to test and evaluate the model.
6. Application step
    - Apply data to the question identified in step 1.
    - Predict output.
    - Evaluate, clean and report results.
    - Obtain insight from the analysis (and visualise it). Here is when we really add value because without a detailed analysis of results, the impact of a ML project is limited. A good practice is the visualisation of results in a way that can be easily interpreted for decision making.

2.2.3 Machine Learning algorithms grouped by learning style

ML can be classified based on its learning style; a popular classification is based on three types of learning systems [180]: supervised learning; unsupervised learning; and reinforcement. Other types have been also proposed: semi-supervised, and deep learning [83, 234].

Below we present a description of each of the learning systems.

Unsupervised learning. Supervised learning is defined as a type of learning where an agent learns patterns via the input even though no explicit feedback is supplied [180]. One of the most common unsupervised learning task is the detection of potentially useful clusters of input samples. In unsupervised learning the input data is not labelled and does not have a known result, the goal is to uncover patterns or structures present in the input data. This may be to extract general rules, to organise data by similarity, or to reduce redundancy. Unsupervised learning is useful in cases where the aim is to discover implicit relationships in a given unlabelled dataset. Examples problems are clustering, dimensionality reduction, and associate rule discovery [180].

Supervised learning. In supervised learning, the agent observes some example input-output pairs and learns a function that maps from input to output. In this case the output value is available directly from the agents percepts (after the fact); the environment is the teacher [180]. In supervised learning the input data is called training data and has a know label or output, the goal in supervised learning is to learn a function that maps inputs to outputs. The ML model is trained in a process that continues until the model achieves a desired level of accuracy on the training data. Supervised learning is useful in cases where a property (label) is available for a certain dataset (training set), but is missing and needs to be predicted for other instances. Example problems are classification and regression.

Semi-supervised learning. Semi-supervised learning falls between supervised and unsupervised learning, which utilises a small amount of labelled data (input-output pairs) and a large amount of unlabelled to train a model [83]. The goal of semi-supervised learning is similar to supervised one except that it learns from both labelled and unlabelled data. Example problems are classification and regression [180]. Semi-supervised learning makes use of labelled and unlabelled data to learn [234]. In [234], the authors divided semi-supervised classification algorithms into three groups: algorithms based on clustering hypothesis [210] as cited in [234], co-training algorithms [17] as cited in [234], and algorithms based on the regularisation framework of graphs [13] as cited in [234].

Reinforcement learning. In reinforcement learning, an agent learns from a series of reinforcements - rewards or punishments, and it is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it [180]. The agent (or learner) receives training information provided by the environment (external trainer) in a form of a scalar reinforcement signal that constitutes a measure of how well the system operates [118]. The agent is not told which actions to take, but rather must discover which actions yield the best reward, by trying each action in turn [118]. In [190], the author defined reinforcement learning as learning to map situations to actions so as to maximise a numerical reward, and indicated that without knowing which actions to take, the learner must discover which actions yield the most reward by trying them and compares the reinforcement learning algorithm with self tuning regulator (STR) from an adaptive control [225]. The purpose of STR (also known as auto-tuning system) is to control systems with unknown but constant parameters, or slowly varying parameters [7]. The algorithm measures the states of the system as outputs, estimate value functions, and output actions [190]. Reinforcement learning is given feedback on its previous experiences but the feedback is rewards or punishments associated with actions instead of desired output or explicit correction of sub-optimal actions. It simulates the human learning based on trial and error. In [86], the reward and punishment based learning are defined as concepts inspired by reinforcement learning.

Deep learning. In [83], Deep learning (DL) is defined as a branch of neural networks, which simulates the multi-layer cognition of human brain to obtain high level features and distributed data structure. DL is based on the idea of firstly using unsupervised learning methods to pre-train the network layer by layer, and then making the training output of lower level as the input for the upper layer, and finally adopting supervised methods to fine-tune the parameters of the whole network [93, 16] as cited in [83].

2.2.4 Supervised learning and classification problems

Supervised learning task can be defined as follows [180]:

Given a *training set* $S = (x_1, y_1), (x_1, y_1), ..., (x_m, y_m)$ of m examples input-output pairs, where each instance x_i belongs to a domain X, and where each y_i is generated by an unknown function $y = f(x)$. A *hypothesis* h is a function that approximates the true function f. *Learning* is defined as a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set. A test set of examples that are distinct from the training set is used to measure

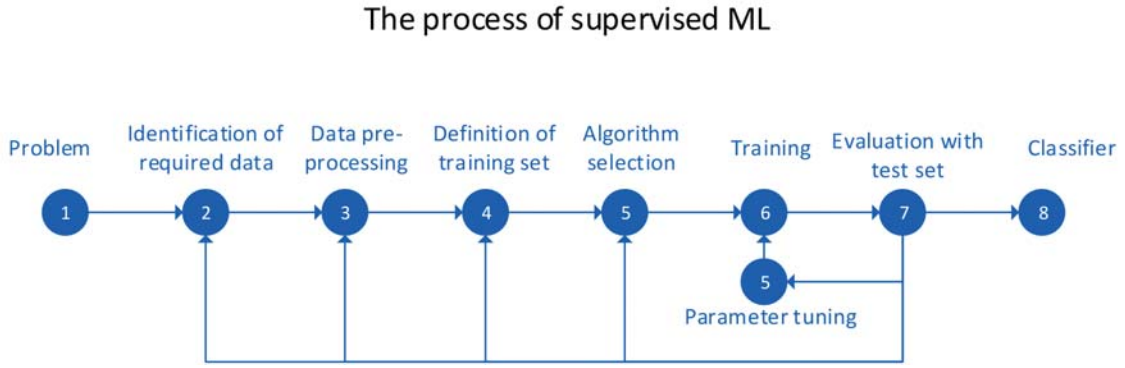

Figure 2.8: The process of supervised Machine Learning. Adapted from [118].

the accuracy of a hypothesis [180]. Finally, the hypothesis *generalises* if it correctly predicts the value of y for novel examples.

Classification is defined as the problem when the output y is one of a finite set of values [180]. Boolean, binary or single-label classification is defined as the classification problem where the output y has only two values [180]. In [91], single-label classification is defined as a problem where each training instance is associated with only one class or label. In single-label classification, the task is to learn some target function $f : X \rightarrow L$ that predicts the correct label for each new instance.

As noted, classification problems are part of the supervised learning, where a label is available for training set but missing and needs for other instances.

Inductive ML is defined as the process of learning a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalise from new instances [118]. Figure 2.8 shows a process of applying ML proposed by [118].

2.2.4.1 Multiclass classifiers

Multiclass classification or categorisation is defined as the problem of assign labels to instances where the labels are drawn from a finite set of labels. A general approach to solve this problem is to reduce the multiclass problem to a set of multiple binary classification problems [40].

Using the definition of classification problem presented at the Section 2.2.4, a multiclass classifier is a function $H : X \rightarrow Y$ that maps an instance x into an element y of Y, where $f : T \rightarrow 1, 2, ..., n$, with $n \in \mathbb{N}$, and $y_1, y_2, ..., y_n \quad (y_j \in T, 1 \leq j \leq n)$.

Multiclass classifiers have been applied to a significant number of problems. In [193], a single-label and a multiclass classifiers were used to estimate the thickeness of melanoma before surgery, through the analysis of computational dermoscopic images.

2.2.4.2 Multilabel classifiers

Multilabel classification (MLC) is defined as the problem where each training instance can be associated with more than one label [91]. The task is to learn some target function $f : X \rightarrow 2^L$ that predicts the correct set of labels (of unknown size) for each new instance. Multilabel learning studies the problem where each example is represented by a single instance while associated with a set of labels simultaneously [233].

Existing methods for multilabel classification follow two main strategies: *problem transformation methods* and *algorithm adaptation methods* [212] [233]. Zhang [233] defined the problem transformation methods as a category of algorithms that tackle multilabel learning problems by transforming it into other well-established learning scenarios (e.g. single-label classification, label ranking, multiclass classification). Algorithm adaptation methods are defined as a category of algorithms that tackle a multilabel learning problem by adapting popular learning techniques (e.g. lazy learning techniques, decision tree techniques, kernel techniques, and information-theoretic technique) to deal with multilabel data directly [233].

In [91], the authors defined *implicit negativity* as the assumption that each training instance will have all the correct positive labels provided and that any label not listed is negative; an adaptation of the backpropragation algorithm was proposed, as an algorithm that does not assume implicit negativity.

Traditional single-label or multi-class classification problems can be regarded as a degenerated version of multilabel classification if each example is confined to have only one single label [233]. In [233], the authors proposed that the key challenge of learning from multilabel data lies in the overwhelming size of output space because the number of label set grows exponentially as the number of class label increases.

During the past decade, multilabel classification has attracted significant attention from machine learning and related communities and has been widely applied to diverse problems [234]. Multilabel classification has been applied in many sectors, e.g. in [106], MLC was used to classify land cover (physical material at the surface of the earth). For content coding of psychotherapy transcripts [71], for nonintrusive electrical monitoring inside residential buildings load monitoring [9], to label incompletely labelled biomedical text data [113], to recognise surface materials from a photograph

[228], to identify small scale-incidents using information extracted from tweets [189], to support automated checking of regulatory and contractual documents in construction [184], to predict doctor label for medical recommendations in the domain of 5G communication [84], to classify power quality complex disturbances [130], gene expression analysis with small sample size [109], for automatic classification and prediction of drug resistance in HIV patients [92], to predict adverse effects in local analgesia in shoulder arthroscopy surgeries [171], to identify places and classify terrain in visual mobile robot navigation [87], for functional genomics and text categorisation [140], to detect and classify colon cancers based on slide histopathological images [229], for prognosis of breast, lung and prostate cancer [185], and for automatic codification of cancer information service chat transcripts [176].

2.3 System Modelling and Simulation

The Institute for Operations Research and the Management Sciences (INFORMS) [207] defines Operational Research (or Operations Research) (OR) as *the application of scientific and mathematical methods to the study and analysis of problems involving complex systems*, while the Operational Research Society defines OR as *the discipline of applying advanced analytical methods to help make better decisions* [208].

The use of Operational Research as a methodology to improve healthcare systems has increased developed considerably over the years [25, 23]. Simulation, one of the most commonly used Operational Research (OR) techniques, has been extensively used in healthcare, mainly due its ability to deal with variability and uncertainty [25, 23, 167, 143, 179, 63].

In this Thesis, system modelling and simulation is defined as the use of a mathematical or logical simplified representation of a process or system as a basis to perform 'What-If' analysis [25]. Figure 2.9 shows the process of problem solving using simulation. The process begin with a real system to be evaluated; a simplified version of the system is developed using modelling techniques; using this computerised simulation model, the theoretical system is evaluated and different scenarios are simulated; outputs from the simulation are then interpreted based on the real system; once the solution has been translated to the real-system space, the final step is the implementation in the real system. A critical step in the modelling process is the validation of both the conceptual and computerised model [61] [62].

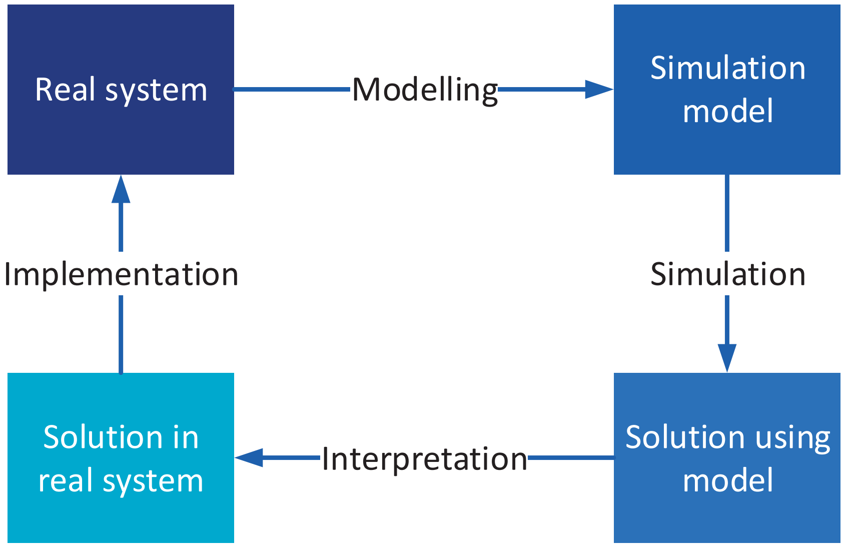

Figure 2.9: Problem solving using simulation.

2.3.1 Classification of simulation models

Simulation models can be classified based on: time dimension (static - dynamic); degree of certainty (deterministic - stochastic); and nature of change (continuous - discrete). Figure 2.10 depicts these classification.

- *Time dimension.* A static simulation model is a representation of a system at a particular point in time (i.e., time plays no role). Dynamic simulation model is a representation of a system as it change over time.
- *Degree of certainty.* A stochastic simulation models is the representation of a system containing at least one random variable as input (with certain probability). A deterministic simulation model is a representation of a system that contain no random variables and no degree of randomness.
- *Nature of change.* Discrete simulation model represents a system where the state variables change only at a discrete points of time. Continuous simulation assumes that states variables change continuously over time.

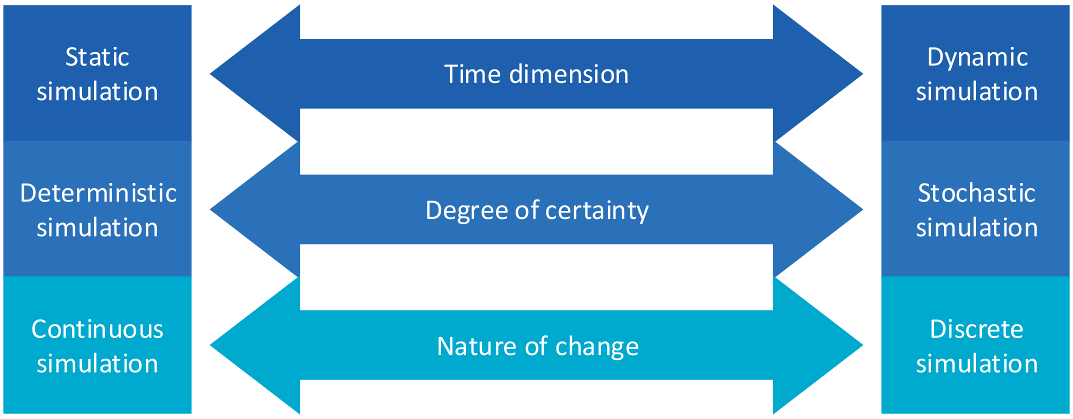

Figure 2.10: Classification of simulation models.

2.3.2 Discrete Event Simulation

Discrete Event Simulation (DES) is one of the methodologies used in system modelling and simulation, other types are System Dynamic, Agent Based simulation, Behavioural simulation, and Monte Carlo simulation [217, 59, 167]. Figure 2.11 shows the relation between DES and other common modelling methodologies [25, 167].

DES can be defined as a stochastic OR technique, used to analyse specific processes and use of resources, where the system can be modelled as a network of queues and activities, with states changing at discrete points in time, and objects with individual characteristics that define what happens within the simulation [177].

The key concepts of DES are entities, attributes, events, resources, queues, processes, and time [107]. Entities (basic object for the simulation) are object that have attributes, experience events, consume resources, and enter queues, over time; Attributes are features specific to each entity that allow it to carry information (e.g., age, sex, health status, past events, etc.). These attributes may change during the simulation, and are used to determine the response of an entity to events in the simulation; An events comprises a specific change in the state of an entity or the system at a specific point in time; Resources are objects that provide service to an entity. Resource availability is typically associate to constrains in the system, and restricting the flow of entities through the system; Queues are formed when an entity requires a non available resource; A process allows an entity to execute a series of methods in simulated time while other entities execute their own series of methods. Processes

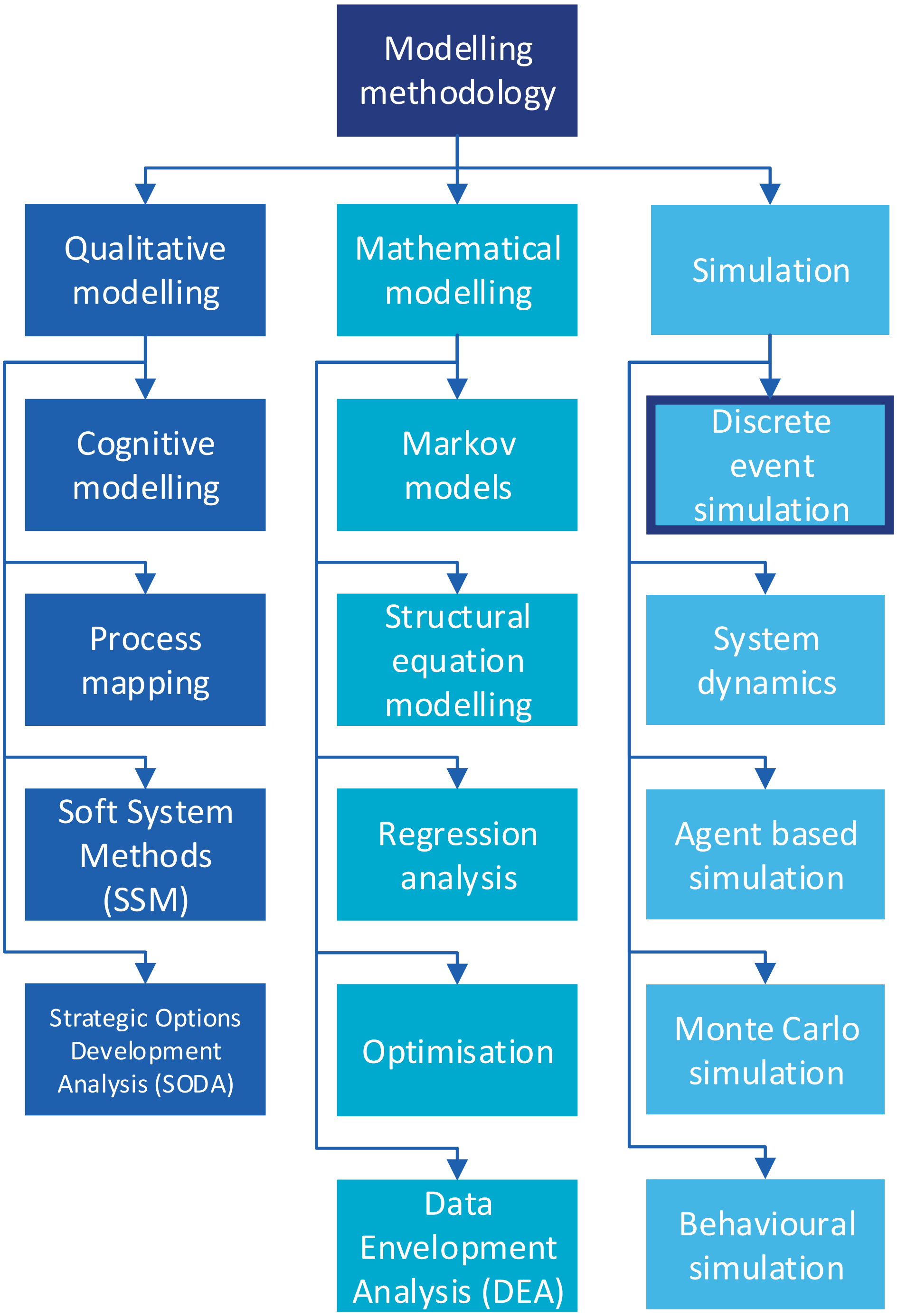

Figure 2.11: Discrete Event Simulation and other modelling methodologies.

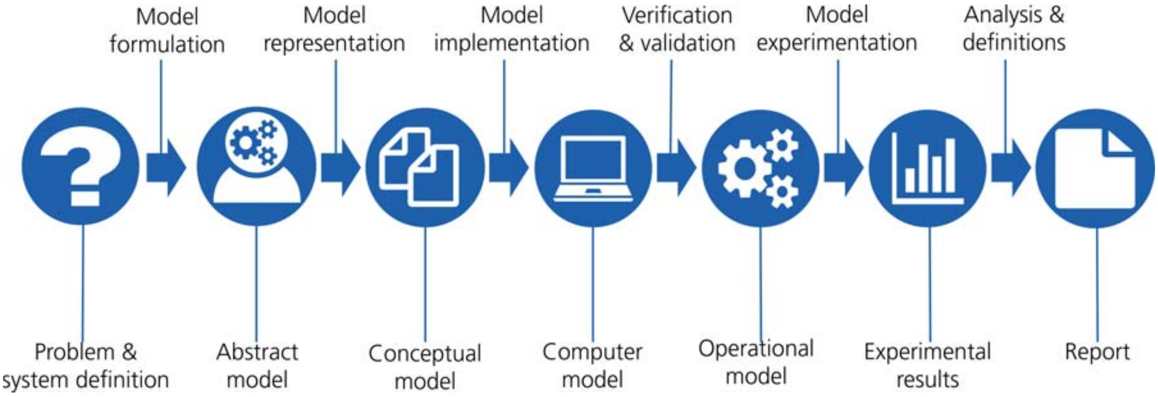

Figure 2.12: Stages of DES modelling process.

are associated to resources consumption; Finally, a simulation clock (initiated at the start of the model run) is crucial to keep track of current simulation time.

A DES can be described using a series of logical steps. Accepted stages of the modelling process are: structural development, parameter estimation, model implementation, model analysis, and representation and reporting [107].

Figure 2.12 shows the different stages of the DES process, including main activities and outcomes. This definition assumes that a DES implementation starts by defining the problem and system to be represented, including the components described above; Next step is the development of an abstract and simplified model of the system; In the conceptual modelling, both the organization of model-components and the identification of behaviour and system control is performed [69]; The task of model implementation involves programming the conceptual modelling and the population of the model with input parameters. In this definition, the parameter estimation assumes that all the data are available, or that expert elicited data can be applied to complete missing information; In the analysis step, the simulation model is used to perform 'What-If' analysis; Final step is the reporting of both the simulation model, and results and findings.

Nevertheless, many simulation projects are impeded to be properly evaluated and implemented by lack of suitable data [21, 200, 24]. We recognise the value of information and data, and its impact on application-oriented DES projects supporting decision making process. For this reason, we proposed additional stages to the DES modelling process. Figure 2.13 includes activities and outcomes related with data collection, preprocessing, and modelling. In the next section, we will describe this steps as part of an Social Network Analysis.

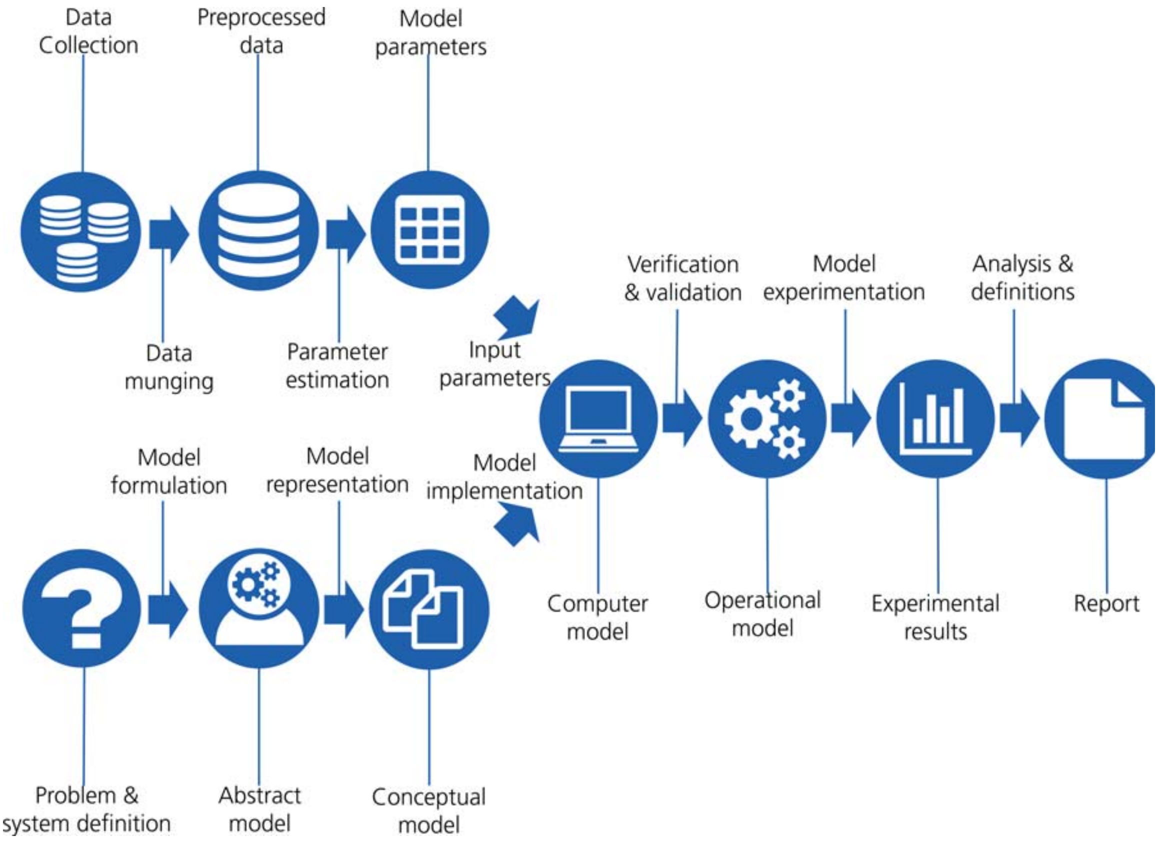

Figure 2.13: Proposed additional steps in the DES process.

2.4 Data Analytics

This section presents a review of data analytics and a review of Social Network Analysis from the data-analytic point of view.

2.4.1 Data Science and analytics

The concept of data was defined by Gould ([81] as cited in [150]) as *a representation of facts or ideas in a formalised manner capable of being communicated or manipulated by some process*. The term *Data Science* was coined by Cleveland in 2001 [36] and argued that *Data Science is generally involving the mixture of statistics and large-scale computing*. Since then many definitions have been presented. Naur [150], for example, defined Data Science as *the science of dealing with data, once they have been established, while the relation of data to what they relation of data to what they represent is delegated to other fields and sciences*. In [34] the term Data Science is defined as *a fusion of computer science and statistics*. The author suggested that from the statistics research point of view, both statistics and data science have the same objectives, but data science focuses more on the volume and the variety of data. While from the perspective of computer science research, data science is more practical.

The relationship between Data Science and Statistics has motivated a big academic debate [27, 45, 80, 231, 53]. In 2015, as part of the John Turkey 100th birthday celebration, Donoho [50] presented the paper '50 years of Data Science', relating the origin of Data Science with the Tukey's work [213], where the author identified (50 years ago) four major influences on data analysis: formal theories of statistics, developments in computers, increasing data availability, and the emphasis on quantification across different disciplines. Donoho argued that the current popularity of Data Science is motived by commercial interests rather than intellectual developments, and concluded that Data Science is, in fact, an "enlargement of academic statistics and machine learning".

The connection between Data Science and Operational Research has also generated academic discussion (see [222, 178]). In [127], the author presents the results of a INFORMS members survey, where the 93% of the respondents indicated a relationship between Operational Research and Data Analytics, and identified three main points of view: Data Analytics is a subset of Operational Research, Operational Research is a subset of Data Analytics, and Advanced Analytics is the intersection between the two fields, with similar results in the survey (29%, 30%, and 28%, respectively). A sign of the close relationship between Operational Research and Data Science are the recently changes in INFORMS and The OR Society, the two largest societies in the field of Operational Research. INFORMS changed from *The Institute for Operations Research and the Management Sciences* to *The Institute for Operations Research, Management Sciences, and Analytics*, while The OR Society is defined as *The OR Society - Operational Research, at the heart of analytics* in its website.

2.4.2 Social Network Analysis (SNA)

Social Network Analysis (SNA) is becoming an important tool for researchers and managers. This section presents an overview of the basic concepts of SNA in data analysis including SNA metrics and performances.

2.4.2.1 Social Network Analysis

Network studies is a topic that has gained increasing importance in recent years [160, 209] . In [134], Social Network is defined as a set of socially relevant nodes connected by one or more relations. Social Networks are formally defined as a set of nodes (or network members) that are tied by one or more types of relations [219]. Nodes, or network members (commonly persons or organisations, but in principle any units that can be connected to other units can be studied as nodes, e.g. groups, subunits of organisations, or more macro-level human systems [42]), are the basic units in a social network.

SNA is an interdisciplinary technique developed under many influences, the most important coming from mathematics and computer science [160, 85]. The study of SNA started from around 1900, and originating mostly in the research areas of sociology [209]. During this period, the studies of SNA focused on small groups and small social networks. However, computer science have become very important for SNA and the field itself is moving from sociology to computer science [192].

SNA has been described as a perspective, paradigm, or a strategy for investigating social structures, rather than a formal theory or a methodology [134, 160]. SNA takes as its starting point the premise that social life is created primarily and most importantly by relations and the patterns formed by these relations [134]. The institutionalisation of SNA began with the foundation in 1978 of the International Network for Social Network Analysis (INSNA) [160]. In 1988, one of the earliest texts dealing exclusively with SNA was published [112].

Because SNA consider the networks to be the primary building blocks of the social world, they not only collect unique types of data, but they begin their analyses from a fundamentally different perspective than that adopted by researchers drawing on individualist or attribute-based perspectives [134]. In SNA, the relationships between actors become the first priority, and individual properties are only secondary. In the traditional individualistic social theory and data analysis, properties of actors are the prime concern, and individual actors are considered to be making choices without taking the behaviour of other into consideration [160]. In [134], the author argued that network explanations do not assume that environments, attributes or circumstances affect actors independently, do not assume the existence of uniformly cohesive and discretely bounded groups, and relations themselves are often analysed in the context of other relations.

Up to now there has been no commonly accepted definition for the term Social Network Analysis. It is just in the last few years that some researchers in the field tried to propose a definition. In [170], SNA was described as *the contextualisation of social structures as a network with ties connecting members and channelling resources, focuses on the characteristics of ties rather than on the characteristics of the* *individual members*, and the author suggested that communities are views as 'personal communities', i.e., as networks of individual relations that people foster, maintain, and use in the course of their daily lives.

In this thesis, we adopt the definition presented in [191]:

Definition 2.4.1. Social Network Analysis conceptualises individuals or groups as 'points' and their relations to each other as 'lines'. It is concerned with the patterns formed by the points and lines and involves exploring these patterns, mathematically or visually, in order to assess their effects on the individuals and organisations that are the members of the 'networks' formed by the intersecting lines that connect them. It therefore takes the metaphorical idea of interaction as forming a network of connections and gives this idea a more formal representation ir order to model structures of social relations. Treating a social structure as a network is the cornerstone of SNA.

2.4.2.2 Relation between Social Network Analysis and other fields

From the sociology point of view, sociometrics can be described as the origin of SNA, with two main schools: Harvard School (W. Lloyd Warner), and the Manchester anthropological school [160].

From the Computer Science point of view, the origins of SNA are related with graph theory (study of graphs, which are mathematical structures used to model pairwise relations between objects). The paper written by Euler on the Seven Bridges of Königsberg and published in 1741 is regarded as the first paper in the history of graph theory [57]. A graph is made up of vertices, nodes, or points where are connected by edges, arcs, or lines [209].

The first book in networks was published in 1936 by D. König [111], since then, Networks have been used for modelling different systems of the real world, mainly using the concepts of *complex networks* (small-world networks and scale-free networks), where nodes represent the different constituents of the system and edges depict the interactions between them [19]. Many of this systems (including transport networks, communication networks, biochemical networks, social networks, infraestructure networks, and some others) are know to have common characteristics in their behaviour and structure [33].

2.4.2.3 Social Network Analysis process

Despite the many types of analysis that can be performed as part of a SNA, there is a common set of key steps including the identification of goals of the analysis and

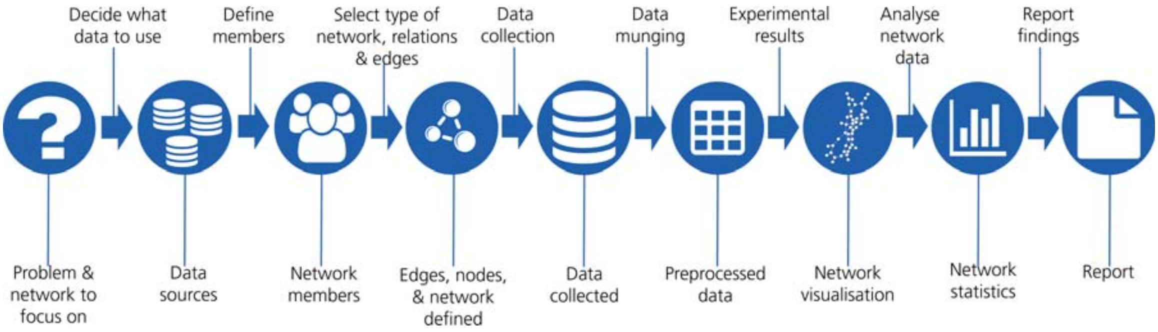

Figure 2.14: Stages of Social Network Analysis process

research questions, gathering data, and visualising and analysing the data. Figure 2.14 shows the stages in a Social Network Analysis process.

1. First step in a SNA process is to identify the problem and social network to focus on, developing an understanding of the application domain. SNA can be used to accomplish a variety of high level goals, each of which includes a large number of potential subgoals and research questions, while these questions vary considerably, they all focus on understanding social structures and how those influence outcomes of interest.

2. Once the question and social network are identified, next step is to decide what data to use. Table 2.1 summarises main social network data sources. Data source that require more effort typically allow for more flexibility in the specific types of data that are collected.

3. After that, network members must be defined. Defining which members to include in a network analysis often poses an early challenge [134]. Three approaches to addressing this problem are:

   - a position-based approach those actors who are members of an organisation or hold particular formally defined positions to be network members and all others would be excluded.
   - an event based approach to defining the boundaries of network looks at who had participated in key events believed to define the population.
   - a relation-based approach begins with a small set of nodes deemed to be within the population of interest and then expands to include other sharing particular types of relations with those seed nodes as well as with any nodes previously added.

4. After identified network members, the next step is to select the type of network, define what an edge will represent. These could include collaborations, friendships, citations, resource flows, information flows, emails, or any other possible connection between these particular units [219]. Relations between individuals can be measured as *directed* or *indirected* and as *binary* or *valued* [134].

5. Once the networks and relations are defined, the next step is te data collection. Depending on the specific data needs, this process may take considerable effort. Network data can be collected through observation, from archives and historical material, from records from electronic communication, etc. [134].

6. Following the data collection, a data pre-process (data munging or data wrangling in the data science community) may be required. As will be discussed trough this Thesis, the complexity of this process is usually underestimated in many data driven applications, and depending on the context and data, can be the most time and resources consuming part of the project [84] [55].

7. With the preprocessed data, the next stage is to visualise the network using an appropriate tool. SNA requires the use of specialised software designed to compute network metrics and visualise network graphs. Examples of software are: Gephi, NetMiner, NodeXL, Pajek, R, Socioviz, etc.). For more details see [187].

8. After data visualisation, an analysis of network, nodes and edges measures and statistics analysis is required. The data are used to calculate measures of the properties of network positions, nodes, edges, clusters, and networks as a whole. Is in this stage when central nodes and communities are identified. A summary of the main network metrics in SNA is presented in the Section 2.4.2.4.

9. The final step is the validation of findings, design of a strategy to improve network outcomes, and report the SNA outcomes.

2.4.2.4 Social Network Analysis metrics

SNA metrics are used to evaluate and compare networks and nodes. SNA metrics can be distinguished into those which evaluate the entire network (aggregate network metrics) and those that only assess a specific node (Node specific metrics).

Aggregate network metrics are a set of metrics used to characterise the network as a whole. This allows to compare different networks, or the development of a

Table 2.1: Main social network data sources

Data source	**Comments**	**Effort level**
Raw data from CRM system	Extract raw data from a customer relationship management (CRM) system.	Medium-high
Phone calls records	Inferring a network from a list of calls.	High
Network survey	Asking people to manually characterise their relationships with other people (e.g., ”list (or select) the people you collaborate the most”). These can be administered via paper or, via specialised online survey tools.	High
Social network Web sites	Extract data from social Web sites such Facebook, Twitter, and Linkedin. Application programming interfaces (APIs), import wizards, plug-ins, or stand-alone network data capture tools.	Medium
Email datasets	Inferring a network from a list of emails.	Easy
Corporate records/registers	Including stakeholders mailing lists, company accounting records, statutory company registers and records (e.g., register of members, directors, shareholders).	Medium-high
Event attendance	Register of attendances to events (e.g., annual stakeholders events, workshops, seminars).	Medium
Co-citations/co-authorship	Register of co-citations or co-authorship (e.g., academic papers, conferences papers/posters, reports).	Medium-high

network over time (when analysing dynamic networks). Visualising entire networks is often useful to identify structures such as the *core* or the *periphery* of a network, network *clusters*, and other patterns. However, many graphs are too large to meaningfully visualise and some properties of a graph are difficult to visualise (e.g., the longest geodesic distance) making aggregate network metrics essential.

Using concepts from complex networks (or graph) theory, a network can be represented as graph, a mathematical object $G = (V, E)$ composed by a set of nodes or vertices $V = \{v_1, \ldots, v_n\}$ that are pairwise joined by links or edges belonging to the set $E = \{e_1, \ldots, e_m\}$, with $E \subseteq V \times V$. The term *complex network*, or simply *network*, often refers to real systems while the term *graph* is generally considered as the mathematical representation of a network [33].

The **adjacent matrix** $A(G) = (a_{ij})$ of a graph $G = (V, E)$ determines the graph completely and is defined by the conditions:

$$a_{ij} = \begin{cases} 1 \text{ if } \{i,j\} \in E \\ 0 \text{ if } \{i,j\} \notin E \end{cases} \tag{2.4}$$

Assuming there are not self-loops, $\forall i \in \{1,\ldots,n\}$ we get that $a_{ii} = 0$.

A **walk** of length k in G is a set of nodes $\{v_1, v_2, \ldots, v_k, v_{k+1}\}$ such that for all $1 \leqslant j \leqslant k$, there is an edge between v_{i_j} and $v_{i_{j+1}}$. A **path** is a walk where $v_{i_1} = v_{i_{k+1}}$. A **cycle** is a path with an edge between the first and last node, i.e., a cycle is a closed path. A **triangle** in G is a cycle of length 3.

The **geodesic distance** $d(v_i, v_j)$ between two nodes is the length of the shortest path connecting v_i and v_j.

The **diameter** of a graph $G = (V, E)$ is defined as:

$$diam(G) = max_{(v_i,v_j)\in V} d(v_i, v_j) \tag{2.5}$$

The **link density** of a network takes values within the interval $[0, 1]$, where $\Delta \ll 1$ represents a *sparse* network, $\Delta \lessapprox 1$ means G is *dense*, and $\Delta = 1$ represents a *complete* or fully connected network of n nodes.

Another popular concept in network topology is the **average path length** (or characteristic path length), defined as:

$$l_G = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k=1,k\neq j}^{n} d(v_j, v_k) = \frac{1}{n(n-1)} \sum_{j\neq k \in V} d(v_j, v_k) \tag{2.6}$$

Average path length can be used to represent the performance of a network (the bigger is l_G the smaller is its performance), and also to represent the concept of network's vulnerability (the bigger is l_G the higher vulnerability in a network). In the context of SNA, vulnerability can be described as the quantification of the network security and stability under the effects of removing a finite number of edges and/or nodes.

Complex graphs arising in real-world applications tend to be highly irregular and exhibit a non-trivial topology - in particular, they are far from being either regular, or completely *random*. Complex networks are very often:

- *Scale-free*, meaning that their degree distribution tends to follow a *power law*: $p(k)$ = number of nodes of degree $k \approx c \cdot k^{-\gamma}, \gamma > 0$. Frequently, $2 \leq \gamma \leq 3$. This implies *sparsity* but also the existence of several highly connected nodes (hubs).

- *Small-world*, meaning that the diameter grows very slowly with the number of N of nodes, e.g., $diam(G) = O(logN), \quad N \to \infty$
- *Highly clustered*, meaning they contain a very large proportion of triangles (unlike random graphs).

Node Specific metrics: If the analysis is focuses on characterise how important an individual is within a particular social network, a node specific metric must be used. There are many different ways that a individual may be important, for example, a individual may be popular, another may be connected to popular people despite having few connections, etc. A set of quantitative measures called *centrality metrics* are defined as part of the SNA, this measures represent the different types of importance in a network.

Degree Centrality C_D is a measure of the number of edges (e) connected to a node (v). For directed networks, the number of incoming links is the in-degree centrality, while the number of outgoing links is the out-degree centrality. Using graph theoretical notation, the *degree* of a node v_j can be represented by the expression:

$$C_D(v_j) = \sum_{i=1}^{n} e(v_i, v_j) \tag{2.7}$$

where $e(v_i, v_j) = 1$ if and only if v_i and v_j are connected by a line, 0 otherwise.

A node with a high degree can be considered as being well connected and a node with a relative low degree can be considered weakly connected. In a social network, C_D can be interpreted as a measure of popularity (e.g., the number of friends one has) [20]. Using a email network as example, in-degree may measure the number of messages one receives, while out-degree may measure the number of messages sent.

The **average degree** of a network, calculated over all the nodes of G can be represented as follow:

$$\langle \kappa \rangle = \frac{1}{n} \sum_{i=1}^{n} C_D(v_i) \tag{2.8}$$

Given a network $G = (V, E)$ where $V = \{1, \ldots, n\}$ and $E = \{e_1, \ldots, e_m\}$, the number of edges of G is at most $\frac{n(n-1)}{2}$.

The link **density** of G represents the amount of interconnectivity in a network and is defined by the equation 2.9:

$$\Delta = \frac{2m}{n(n-1)} \tag{2.9}$$

The **clustering coefficient**, when applied to a single node, is a measure of how complete the neighbourhood of a node is. When applied to an entire network, it is the average clustering over all of the nodes in the network [124] [220].

This metric measures how close the neighbourhood of a specific node v_i is to a complete subgraph, where the neighbourhood of a node is defined as the set of nodes that are immediately adjacent to node v_i.

The local clustering coefficient C_i (defined in [220]) for a node v_i is the proportion of edges between the nodes within its neighbourhood divided by the number of edges that could possibly exist between them, i.e., the number of triangles in which node v_i participates normalised by the maximum possible number of such triangles, where t_i denotes the number of triangles around v_i. C_i is defined as:

$$C_i = \frac{2t_i}{C_D(v_i)(C_D(v_i) - 1)}. \tag{2.10}$$

where t_i is defined as:

$$t_i = \{(i, k) \in E \mid (i, j) \in E \wedge (i, k) \in E\} \tag{2.11}$$

We can refer to the clustering coefficient $C(G)$ of a graph G as the average clustering coefficient of all nodes in the network:

$$C(G) = \frac{1}{n} \sum_{i=1}^{n} C_i \tag{2.12}$$

By definition $0 < C_i \leqslant 1$ and $0 < C(G) \leqslant 1$.

Eigenvector centrality is a concept part of the spectral graph theory [35, 115], introduced to find out the critical and most influential nodes in a network [20]. It assigns relative scores to all nodes in the network based on the concept that connections to high scoring nodes contribute more to the score of the node than equal connections to low-scoring nodes. Eigenvector centrality can be defined as a measure of a node's importance that considers the importance of the node's neighbours, where importance is calculated as a weighted sum of direct connections and indirect connections of every length, so eigenvector centrality is defined in a circular manner. In the SNA context, an important node (or person) is characterised by its connectivity to other important nodes (or people). A node with a high centrality value is a well-connected node and has a dominant influence on the surrounding network.

Google utilises a variant of the eigenvector centrality (PageRank method) as a global ranking of all web pages, condensing every page on the World Wide Web into a single number based on their location in the Web's graph structure. PageRank is one of the algorithms (and was the first) used by the company to order search results so that more important and central Web pages are given preference.

Betweenness Centrality is another approach to the study of network's centrality. This is based on the idea that critical nodes and edges stand between others, playing the role of an intermediary in the interactions or in the communications. A node in a communication network is central to the extent that it falls on the shortest path between pairs of other nodes. Betweenness centrality quantifies the number of occurrences of the node as a bridge along the shortest path between two other nodes. The number of shortest paths between all other nodes that a particular node is on - i.e., how often a node lies 'between' other nodes [66]. The node betweenness centrality $B(G)$ of a network G is [66]:

$$B(G) = \left(\frac{1}{n} \sum_{i \in V} b_i \right). \tag{2.13}$$

where b_i is the betweenness of the node $i \in V$ given by:

$$b_i = \frac{1}{n(n-1)} \sum_{k,j \in V_{i \neq j}} \frac{n_{kj}(i)}{n_{kj}}. \tag{2.14}$$

where n_{kj} is the number of different geodesics that join k and j, and $n_{kj}(i)$ is the number of geodesics that join k and j passing through i.

Closeness centrality (or closeness) [182, 10, 67] is a measure of the degree with which a node is nearer (closeness) to the rest of the nodes on a network, either directly of indirectly. Closeness reflects the ability of an individual to access information through the network members. Normalised closeness centrality (or closeness) of a node is defined as the average length of the shortest distance (geodesic distance) between the node and all other nodes in the network. Thus the more central a node is, the closer it is to all other nodes. Closeness centrality $C(G)$ can be represented mathematically as:

$$C(G) = \frac{1}{\sum_i d(v_i, v_j)}. \tag{2.15}$$

where $d(v_i, v_j)$ is the distance between nodes v_i and v_j.

Harmonic closeness centrality $H(G)$ is used when a graph is not strongly connected. $H(G)$ is calculated using the sum of reciprocal of distances, instead of the reciprocal of the sum of distances, with the convention $\frac{1}{\infty} = 0$.

$$H(G) = \sum_{v_i \neq v_j} \frac{1}{d(v_i, v_j)}. \tag{2.16}$$

Edges can also carry weights to measure the capacity or the intensity of the relationship between two nodes. Examples of this situations are the existence of strong and weak ties in social networks. These networks are better described as *weighted networks*, i.e., networks in which each edge has associated a value measuring the strength of the relationship, and in these cases a so called weighted matrix $W = (w_{ij})$ is defined, whose entry w_{ij} is the weight of the edge from node i to j. A weighted (directed or undirected) network is defined as a triplet $G = (V, E, W)$, where (V, E) is a (directed or indirected) network and $W = (w_{ij})$ is the weights matrix of G.

2.5 Conclusions

This chapter has provided a review of historical and current literature relating to the main data-driven methodologies analysed in the Thesis. Differences and similarities between Metaheuristics, Machine Learning, Discrete Event Simulation, and Social Network Analysis were presented.

A key component in all the different approaches is the importance of the correct definition of the problem to focus on. Another important finding of the review is the important role of the data selection, extraction, pre-processing, and conversion in application-oriented projects.

2.6 Chapter Summary

Section 2.1 introduced the concepts of Metaheuristic Optimisation, Section 2.1.1 presented the fundamentals of optimisation and Combinatorial Optimisation. Section 2.1.2 focused on different approaches for solving optimisation problems, and introduced the concepts of computational complexity theory. Section 2.1.3 presented the history of metaheuristics and discussed different definitions. Section 2.1.4 summarised different optimisation approaches, and described Genetic Algorithms.

Section 2.2 introduced Machine Learning (ML), discussed different definitions found in the literature, and discussed the difference between ML and the traditional programming paradigm. Section 2.2.1 described the ML process. In Section 2.2.2 we proposed a new framework for ML applications. Section 2.2.3 presented a classification of ML algorithms based on learning style. Section 2.2.4 introduced supervised learning and classification problem, including Multiclass and Multilabel classifiers.

Section 2.3 reviewed System Modelling and Simulation in the context of Operational Research (OR). Section 2.3.1 introduced problem solving using simulation and presented a classification of simulation models. Section 2.3.2 described Discrete Event Simulation (DES), including the key concepts and components of a DES, and described the DES process. In this section we discussed the impact of the data in the DES project, and we proposed additional stages to the 'standard' DES process.

Section 2.4 introduced the concepts of Data Analytics and Data Science. Section 2.4.2 focussed on Social Network Analysis (SNA). Section 2.4.2.3 described the SNA process. Section 2.4.2.4 presents the most popular metrics used in the analysis of Social Networks, linking SNA with the graph theory.

Metaheuristic Optimisation, with a focus on Genetic Algorithms; a review of Machine Leaning; Modelling and Simulation, with particular attention to Discrete Event Simulation; and finally a review of Data Analytics, focussing on SNA.