Chapter 72
What Is an Algorithm?
Traditional vs. Intelligent Algorithms

Julia de Miguel
The London School of Economics and Political Science, UK

J. Ángel Velázquez-Iturbide
https://orcid.org/0000-0002-9486-8526
Universidad Rey Juan Carlos, Spain

ABSTRACT

Algorithms have emerged in past years as an object of public interest and debate. However, this term is used to name very different conceptions. The chapter presents the different definitions of the term "algorithm," especially the traditional conception of algorithms as used in informatics and the novel conception of (intelligent) algorithms that have emerged in the last years. The main contribution of the chapter is the characterization of both types of algorithms in terms of the problems they are intended to solve. Furthermore, the chapter contributes by analyzing other uses of the word "algorithm" where it is an inadequate term. Finally, the chapter discusses misconceptions on algorithms present in other disciplines, which is an obstacle to mutual understanding and a source of mismatch between the disciplines.

INTRODUCTION

In the last few years, "algorithm" has become a trendy word, even almost a buzzword. In the past, it was almost exclusively limited to the informatics[1] community, whereas it has now been extended to many application areas with high social impact and polarized debates (Burrell, 2016). The main contribution of the chapter is to clarify the features of both types of algorithms. The elements and properties that characterize each kind of algorithm are identified, as well as their similarities and differences. Moreover, the purpose of this chapter is to provide a comprehensive overview of the different meanings of the word "algorithm". Consequently, another contribution of the chapter is to identify several situations where there is an abuse of the word "algorithm" by either communication media or researchers, and where an alternative term would be more informative. Finally, it is also noted that conceptions of algorithms vary within the disciplines.

DOI: 10.4018/978-1-6684-7366-5.ch072

The structure of the chapter is as follows. The background section presents a short historical introduction to algorithms. The focus section contains the bulk of the chapter, with a detailed analysis of different types of algorithms and non-algorithms, and an emphasis on "traditional" algorithms and the emerging "intelligent" algorithms. The directions for research identifies the most current research trends, especially related to intelligent algorithms. A short conclusions section closes the chapter.

BACKGROUND

The word "algorithm" seems to derive from the name of the ninth-century Persian mathematician al-Khwarizmi (Knuth, 1985), altered under the influence of the Greek word "arithmos" (i.e., number)[2]. However, the word has a long tradition, first in mathematics and later in informatics. Along the centuries, mathematicians have proposed different methods to solve mathematical problems, which are precise enough to be considered algorithms. Although algorithms can be traced back to the Babylonians (Knuth, 1972), the oldest algorithms with a name are Greek, e.g., the sieve of Eratosthenes or Euclid's algorithm. In subsequent centuries, the mathematicians elaborated many algorithms, mainly calculation formulae (e.g., how to solve a second-order equation) and precise methods (e.g., Gauss-Seidel method to solve systems of linear equations).

However, it was with the advent of computers that algorithms were studied systematically. Furthermore, their scope was widened from mathematical problems to information processing problems, such as sorting the elements in a sequence. Some authors even consider that "*computer science is primarily the study of algorithms*" (Knuth, 1985). Moreover, the possibility of automatically applying the algorithms by means of computers gave rise to some concerns on algorithms that did not exist previously, mainly the analysis of their efficiency (in their use of time and space) and the theoretical issues of computability and tractability.

The high relevance of algorithms within informatics was early reflected on curricula. Thus, the Association for Computing Machinery (ACM) Curriculum 68 (Atchison et al., 1968) had a first course B1, "Introduction to computing", with the following initial descriptor: "*algorithms, programs, and computers*". About twenty years later, the ACM and the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) joined forces to develop curricular recommendations. The Denning report (Denning et al., 1989) was the first tangible outcome of this collaboration, setting the bases for future recommendations. It identified nine subareas of informatics, being "algorithms and data structures" the first one. It is introduced in the following way: "*This area deals with specific classes of problems and their efficient solutions. Fundamental questions include: For given classes of problems, what are the best algorithms? (…) How general are algorithms −i.e., what classes of problems can be dealt with by similar methods?*" This relevant situation did not change in subsequent curricular recommendations for computer science (e.g., ACM & IEEE-CS, 2013).

In parallel, research on algorithms and their properties was acknowledged as one of the fields of informatics for decades. It comprises both theoretical elaborations (notably, the theory of computability and complexity), mathematical and experimental methods to analyze algorithms with respect to clearly defined criteria, a corpus of efficient algorithms for highly relevant problems, and methodical approaches to algorithm design.

FOCUS OF THE ARTICLE

In this section, different types of algorithms are analyzed. First, the traditional definition of algorithm is presented, from the point of view of informatics. To further clarify the meaning of algorithm, the definition is complemented with examples of non-algorithms, with the properties and theoretical implications of algorithms, and with an analysis of relevant classes of algorithms that exhibit some problematic features. Second, the type of intelligent algorithms, lately popularized, are presented and analyzed. Their differing features with respect to traditional algorithms are discussed, as well as their societal consequences. A commentary is also included on the different meanings of algorithm between the researchers and practitioners of different fields of knowledge.

WHAT IS AN ALGORITHM

We may find informal definitions of algorithm in some textbooks (surprisingly, some algorithmic textbooks address the study of algorithms without giving a preliminary definition). Thus, Brassard and Bratley (1996, p. 1) claim that an algorithm "*is simply a set of rules for carrying out some calculation, either by hand or, more usually, on a machine*". Horowitz et al. (1996, p. 1) define an algorithm as "*a finite set of instructions that, if followed, accomplishes a particular task*". Cormen et al. (2009, p. 5) give a more precise definition: "*an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output*". Furthermore, "*we can also view an algorithm as a tool for solving a well-specified computational problem*" (Cormen et al., 2009, p. 5).

It is more clarifying to resort to the characterization of problems and algorithms given by theoreticians (e.g., Manna & Waldinger, 1980). A precise definition of a computational problem is as a transformation of valid input data into output data, with the following elements:

- **Input and output:** Each characterized as a set of values.
- **Pre-condition:** It is a predicate that valid input data must satisfy. The pre-condition allows restricting the set of input data to those for which the problem makes sense.
- **Post-condition:** It is a predicate that declares the relationship that must be held between each specific input value and any corresponding output value.

In other words, given any specific and valid instance of its input data, the algorithm will perform the steps necessary to compute any output value that satisfies the input-output relationship. Typically, many alternative algorithms can solve a given problem.

For instance, consider the problem consisting in calculating the maximum common divisor of two numbers, which also is a number. The pre-condition of the problem states that both numbers must be non-negative, and both cannot be simultaneously zero (in that case, the maximum common divisor would be infinite). The post-condition of the problem states that the result must be the largest of all the common divisors of both numbers. For instance, given numbers 18 and 24, their largest common divisor is 6. The problem can be efficiently solved by Euclid's algorithm, among others.

Problems need not be restricted to numeric values, but they may involve other types of data, such as characters or Booleans, as well as data structures (i.e., structured collections of data), such as graphs with distances or strings of characters. Many popular algorithms address problems with data structures. For instance, quicksort (Hoare, 1961) is an efficient algorithm that sorts a sequence of comparable elements.

An algorithm must fulfill some additional properties (Knuth, 1973; Horowitz et al., 1996) to be considered, as aforementioned, a "*well-defined computational procedure*" (Cormen et al., 2009, p. 5):

- **Definiteness:** Each step of an algorithm is clear and unambiguous. A consequence of the definiteness property is that an algorithm always will produce the same result for the same input data. The algorithm can be stated by any means that describe it without ambiguity, including natural language, and can even be wired in a hardware design. More typically, algorithms are described by means of control-flow diagrams, programming languages or pseudocode. Actually, most algorithm textbooks use pseudocode rather than any real programming language (see e.g., Brassard & Bratley, 1996; Cormen et al., 2009; Horowitz et al., 1997; Kleinberg & Tardos, 2006).
- **Effectiveness:** An algorithm is typically coded for its use with a computer, but a person should also be able to apply it with pencil and paper. In other words, the description of an algorithm should allow its automated execution by an active agent, either by a machine or by a compliant person who does not take any extra decision: "*an algorithm is not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgment*" (Denning, 2017).
- **Finiteness:** The algorithm must terminate after a finite number of steps, giving an answer to the computational problem. Note that the sequential character of an algorithm only refers to its application, not to its description. Actually, only simple algorithms (e.g., some mathematical formulae) are described as a sequence of steps. In general, any algorithm can be stated by means of three control elements (Böhm & Jacopini, 1966): sequence of operations, alternative operations and repetition of operations. In addition, repetition can be expressed either iteratively or recursively.

Note that we may speak in informatics about these issues at different degrees of detail, i.e., at four levels of abstraction (Perrenet et al., 2005): problem, algorithm, program, and execution. The most abstract concern is the problem to be solved. More specific is any procedure that solves the problem, that is, an algorithm. The algorithm may be made even more specific by coding it in a programming language. Finally, that program may be executed for specific input data.

WHAT IS NOT AN ALGORITHM

In different contexts, the word algorithm is often used incorrectly, because at least one of the previous features of algorithms is not satisfied. Below, examples of non-algorithms are critically examined to further clarify what is an algorithm.

A common mistake is to consider algorithms to procedures that contain ambiguous statements, which is against the definiteness property. For instance, cooking recipes are frequently used as an appealing and familiar metaphor to introduce algorithms. Educators should be aware and warn students of the limits (and inexactitudes) of their metaphors (Forišek & Steinová, 2012), but the recipe metaphor is adopted literally. As Brassard and Bratley (1996, p. 2) explain, recipes do not satisfy the definiteness property because there are plenty of ambiguous steps, e.g., "*add salt to taste*" or "*cook until tender*". Similarly, Hill (2016, p. 49) claims that "*if a set of instructions can be said to be followed well or to be followed badly, it is not an algorithm*" (Hill, 2016, p. 49).

This wrong use of the term algorithm also extends to other daily activities, such as "*changing a tire and shucking peas*" (Hill, 2016, p. 49). Denning et al. (2017, p. 33) have warned about this source of misconceptions: "*thus most «human executable recipes» cannot be implemented by a machine. This misconception actually leads people to misunderstand algorithms and therefore overestimate what a machine can do*".

The effectiveness property assumes that an algorithm can be carried out by a compliant person or a computer. In the latter case, the algorithm is coded in a programming language, and we have a program. Here, a confusion between program and algorithm frequently arises. Any algorithm can be written in a programming language, but any program is not an algorithm. To be considered an algorithm, a program must satisfy the other properties of algorithms, in particular the existence of a problem to solve.

For example, consider the classical C program that prints "hello, world" (Kernighan & Ritchie, 1978). This piece of code is a program, but it is not an algorithm. Certainly, the program does perform a task but it does not solve any problem. Therefore, we cannot speak of an algorithm for those pieces of programs that perform mundane computational tasks, such as handling files, rather than solving a problem. Conversely, programs usually include many algorithms, sometimes hidden to the user but useful for certain tasks. For instance, in a word processor, an algorithm automatically distributes the words of a paragraph into different rows to ease the user's task of typing and reading his/her text.

This conflation only could make sense as a conscious decision in a context where the distinction of programs and algorithms is irrelevant. For instance, consider the case of measuring performance: "*We take the wide view and use the term «algorithm» to mean «algorithm» or program» from here on*" (McGeoch, 2012, p. ix). Similarly, courses and textbooks on programming methodology typically refer to programs (Dijkstra & Feijen, 1988), but they actually address the derivation or verification of algorithms.

The confusion between program and algorithm is common within the stream called computational thinking (Wing, 2006). The last two decades have seen the rise of this trend, which advocates for the incorporation of informatics education into the pre-college curriculum. These advocacies often refer to algorithms, but it is difficult to find any definition of what is understood by this term. For example, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) (2011) advocate for "*automating solutions through algorithmic thinking (a series of ordered steps)*". In many computational thinking articles, algorithms are reduced to any ordered steps, without any characterization of the kind of problems they are intended to solve.

This confusion is frequent in the context of block-based programming (Bau et al., 2017), a common approach to programming in primary or secondary education. These languages provide an excellent opportunity to introduce students to programming concepts and skills, while they construct appealing programs. However, the use of these languages usually is not tied to designing or implementing algorithms but to developing open projects, where students creatively construct multimedia animations, games, presentations, etc.

A simple way of checking whether a program embodies an algorithm is to ask its purpose. If a computational problem can be stated, then the program is solving this problem and consequently it embodies an algorithm. If the only sensible summary is either an open task or a transcription into natural language of the program statements, then we just have a program. Actually, translating the behavior of several characters, described in natural language, into a program is a common exercise in block-based programming (Velázquez-Iturbide, 2023).

The confusion between programs and algorithms can also be found in many application areas which are increasingly digitized. In these contexts, the adoption of computers and software to accomplish traditionally manual tasks are often labeled with the term algorithm. A good representative area is music, where the term "algorithmic composition" is used (Edwards, 2011). Here, musicians compose music with the support of specific software and notations (i.e., languages). Thus, there is no problem to solve, but the open problem of composing music. It would be more appropriate to speak about computer-supported music composition, making an analogy of other areas where prefixes such as "computer-supported" or "computer-aided" was incorporated, e.g., Computer-Supported Collaborative Work/Computer-Supported Collaborative Learning (CSCW/CSCL) or Computer-Aided Design/Computer-Aided Manufacturing (CAD/CAM). Note that we do not speak here about generative music composition, based on artificial intelligence, which will be mentioned below.

PROPERTIES AND FACTS ON ALGORITHMS

The topics typically addressed on learning algorithms are out of the scope of this chapter. However, we outline two concerns to better appreciate the implications of the definition of algorithms given above, in contrast with the second type of algorithms to be examined below.

Firstly, an important concern is the properties of algorithms, the most important being:

- **Correctness:** It is an obvious property. An algorithm only is useful if it does solve the problem that it is intended to solve. In other words, given a problem specification and an algorithm, the algorithm is correct if and only if it produces, for any input data that satisfies the problem precondition, any output that satisfies the post-condition.
- **Efficiency:** It can be measured in terms of either time of execution or the amount of space needed to execute it, being time of execution the most pressing concern. Although experimental gathering of measures is illustrative, it is more informative to conduct a formal analysis, determining a time or space function of the algorithm, and its asymptotic simplification (i.e., the time or space complexity of the algorithm). In addition, the analysis of time complexity allows partitioning algorithms into efficient or inefficient (in technical terms, an efficient algorithm has an associated function time that is asymptotically polynomic on the size of input data).

A second concern are some fundamental theoretical results. Their importance is similar to basic laws of physics, since they identify some limits that will never be surpassed. Thus, there exist intractable problems which are inherently inefficient, i.e., no algorithm will ever exist that will solve those problems efficiently. It does not matter how fast a computer will be, large instances of these problems will never be solved in reasonable time. Even more importantly, there exist non-computable problems, i.e., problems for which no algorithm exists that can solve them. Several mathematicians independently proved this result using different mathematical approaches, being the halting problem by Alan Turing the most famous formulation of this finding.

ELABORATING THE DEFINITION OF ALGORITHM

The definition of algorithm given above demands the clarification of some issues that emerge for particular types of algorithms. A first issue is that for many problems (e.g., maximum common divisor or sorting, mentioned above), one single valid result exists for every specific valid input data. However, the definition of problem does not necessarily restrict valid results to just one. Instead, it claims that any result that satisfies the post-condition is valid, which allows the existence of multiple valid results. This situation is evident in combinatorial problems (e.g., the n-queens problem, see Brassard & Bratley, 1996; Horowitz et al., 1997). Another case is found in numeric methods. These algorithms compute approximations to the exact solution of mathematical problems within a given error margin (e.g., computing the value of a definite integral using either the trapezoidal or Simpson's rule). Therefore, the many real approximations to the exact solution that fall within that error margin are valid results.

Some design techniques that solve well-defined problems may look problematic to satisfy the definition and additional properties declared above for algorithms. Therefore, such a definition must be relaxed to make room for them:

- **Definition of problem:** The post-condition states the properties that solutions to a given problem must satisfy to be considered valid. However, the specification of optimization problems includes a second component in their post-condition, i.e., an objective function which declares a measure to optimize. Therefore, any result must be valid and optimal.

The inherent inefficiency of many optimization problems has led to accepting, in many situations, algorithms that only satisfy partially the post-condition, by computing results that are valid but may be suboptimal. These inexact algorithms are called heuristics (Brassard & Bratley, 1996).

The solutions computed by heuristics are assessed by a new property, namely the solution quality (or optimality) of the algorithm (McGeoch, 2012). The property is measured by either an absolute or relative error (also called the approximation ratio).

- **Definiteness property:** Most algorithms have a deterministic behavior. However, probabilistic (or randomized) algorithms are a pragmatic alternative to consider in different situations (Brassard & Bratley, 1996; Kleinberg & Tardos, 2006). These algorithms contain some steps where a key decision is randomly made instead of searching for the best one. Accordingly, the time efficiency of a probabilistic algorithm is measured differently, by its expected time (Brassard & Bratley, 1996). The algorithm may yield different results for the same input data or even terminate without a result. In some cases, different results may be acceptable, as in optimization problems; other undesirable behaviors, such as termination without a result, can be removed by using several techniques (Brassard & Bratley, 1996).
- **Effectiveness property:** When the active agent in charge of applying an algorithm is a computer, the most obvious choice is sequential execution of instructions, but parallel execution is an alternative for multiprocessor computers (Brassard & Bratley, 1996; Cormen et al., 2009; Horowitz et al., 1997).

- **Finiteness property:** In our presentation, we assumed that all input is available from the beginning, but there also exist optimization problems where the input is presented with one request arriving at a time. On receiving a request, the algorithm must make some irrevocable decision, generally without any knowledge of future requests, attempting to optimize some global objective function (Kleinberg & Tardos, 2006). These algorithms are called on-line algorithms (Boyar et al., 2017) and they are usually integrated in computer programs that never end, e.g., operating systems. These algorithms also are assessed with respect to their solution quality, but with different measures than heuristics (Boyar et al., 2017), being the competitive ratio the most frequently used.

A SECOND TYPE OF ALGORITHMS: INTELLIGENT ALGORITHMS

The term algorithm has recently achieved popularity in fields where the computer intends to perform tasks that are non-computable and which, therefore, were traditionally considered more adequate for humans than for computers. Well-known examples are finding relevant information (e.g., web search engines), making suggestions (e.g., recommender systems) or making decisions (e.g., decision-making systems). Some novel algorithms have achieved popularity, for instance, the PageRank algorithm (Brin & Page, 1998) used to rank web pages in Google.

The popularity of and the concern with these algorithms emerged because they often affect directly different aspects of our lives, from e-commerce recommendations on different products to decisions on health, job or bank loans to predictions on education drop-out. Moreover, they have appeared at a historic moment when digitalization of all the social and economic sectors is accelerating. Consequently, they have been called "*public relevance algorithms*" (Gillespie, 2014).

These algorithms are intended to solve not-computable but "human problems", and solving them requires "intelligence". Therefore, we depart from the informatics field of algorithmics and enter into another field of informatics, namely artificial intelligence. Furthermore, given their high complexity, we should speak of systems (or "*algorithmic systems*", see Seaver, 2019) rather than algorithms. This distinction was more apparent in the eighties of the past century, when non-computable problems were addressed by intelligent systems called "expert systems", acknowledging that they performed tasks akin to experts. However, the term algorithm has been popularized and adopted, even in academic circles. Consequently, we will speak of "intelligent" algorithms to differentiate them from the traditional (or classic, see Hill, 2016) algorithms analyzed above.

Let us check whether intelligent algorithms satisfy the definition stated above. For concreteness, let us analyze web search engines. A search engine is a web application that gets an input from the user; for simplicity, we may consider that the input is just a list of words, although web browsers allow users to ask structured queries. In response, the search engine delivers a list of web directions (i.e., URLs), sorted in order of relevance to the user. Certainly, the output set can be clearly identified (i.e., a sorted list of web directions), but it is not the case of the other elements of the search problem:

- **Input:** From the user's point of view, the input set is clearly identified: his/her list of words. However, the search engine cannot answer without information on the web pages published all over the world. Accordingly, a search engine periodically explores the web (by means of specialized programs called crawlers) to locate, analyze and catalog (i.e., indexing) web pages. Given a user query, the engine finds and sorts candidate pages (i.e., ranking) based on their expected relevance to the user. We could argue that the set of all the web pages found was implicit in the user query but, given that crawling is made independently from individual queries, that set changes from time to time. Moreover, search engines try to better satisfy user's needs by using additional data which are not declared in the problem statement, usually without user's awareness, e.g., the user's location and language, or his/her past queries.

- **Post-condition:** The post-condition cannot be stated with precision. In our example, we may informally claim that the user must be satisfied with the usefulness to him/her of the top ranked pages. However, satisfaction and usefulness are subjective constructs, not objective properties that can be operationalized and checked. Explicitly requesting the user additional feedback is not practical. Furthermore, the needs of the same person for the same query may vary from time to time. Consequently, research on search engines experiments with indirect measures of user's satisfaction, such as features of the user's interactions or his/her search sessions (Dan & Davison, 2016).

In summary, the problem addressed by search engines is not well-defined. A similar situation is found in other applications based on intelligent algorithms, e.g., recommender systems (Bobadilla et al., 2013) or decision-making systems (Castelluccia & Le Métayer, 2019). Given the imprecise definition of these problems, intelligent algorithms try to overcome this deficiency from different approaches. Consequently, these algorithms often produce excellent results, even surpassing experts' performance. For instance, well-known techniques to address classification problems include Bayesian classifiers, fuzzy systems, and neural networks, among others (Bobadilla et al., 2013).

FEATURES OF INTELLIGENT ALGORITHMS

Several consequences derive from the lack of a precise definition noted above for the non-computable problems addressed by means of intelligent algorithms. Firstly, the post-condition can only be stated informally, with many answers being potentially valid. For instance, consider recommendation systems. Their main goal is usefulness of recommendations, but it is impossible to define without ambiguity what this means, even in a specific domain, because no single definition of utility works for all users. Thus, the goal of educational recommender systems may vary from suggesting novel or good educational materials to suggesting a peer student or a learning path to predicting a student's performance (Manouselis et al., 2013). Moreover, experience has shown that other goals are also useful to all or some users, and they even may change along time depending on personal context.

The ill definition of the post-condition implies that that there are no obvious criteria to objectively assess outcomes. For instance, one criterion used in recommender systems is prediction accuracy, similar to solution quality in heuristic algorithms. However, other criteria have also been proposed, such as coverage, ranking, novelty, diversity or fairness, to name a few (Bobadilla et al., 2013; Zangerle & Bauer, 2022). In each application, researchers formalize different operationalizations for defining and assessing algorithms outcomes. However, they are not criteria directly implied by the problem statement, but hypothetical features which are systematically studied.

Secondly, the input data necessary to address these problems with better outcomes is only partially known. Designers of intelligent algorithms must hypothesize and test the most relevant data for the task. Thus, the research explores the relevance of all the information potentially available, sometimes by asking for it explicitly, but often without the user's consent. Some data may be related to the user's behavior, and may even be personal data, which are extracted from the user's computer or digital device (e.g., smartphones or even wearables) or from other external resources (e.g., the user's activity in social networks). This heterogeneous information can involve tens of thousands of different attributes and pieces of information.

Intelligent algorithms are often based on sets of specific input data that illustrate expected behavior. Development of intelligent systems often involves a two-phase process. In a first phase, the system is trained with hundreds or thousands of problem instances to refine the details of the model, eventually leading to the final version of the algorithm. Thus, the performance of these algorithms is highly dependent on the quantity and quality of training data, not on the logic of the algorithm. In a second phase, the model is used to solve problems.

The difference between traditional and intelligent algorithms also affects their scale. Given the clear definition of computational problems, traditional algorithms usually can be expressed in one or several pages, thus they can be understood by individuals with the necessary expertise. However, intelligent systems often comprise multiple algorithms. Even "conventional" systems may make use of both traditional and intelligent algorithms, as reported for Wikipedia (Geiger, 2017, p. 34).

UNEXPECTED CONSEQUENCES OF INTELLIGENT ALGORITHMS

The salient features of intelligent algorithms have technical, personal and social consequences (Abiteboul & Dowek, 2020). Firstly, algorithms may lack transparency. Traditional algorithms are transparent, in the sense that full details of them are given by their designers, including their written formulation, perhaps accompanied by a proof of correctness or optimality and its complexity analysis. Some intelligent algorithms also are transparent. However, other algorithms are based on models with many parameters, whose corresponding weights are unknown. In the case of algorithms trained with a data set, transparency would involve knowing the data set used for training.

A related concern is explainability, i.e., the capability of the system to explain the choices adopted by the algorithm to obtain their final outcome. Some models (most notably, neural networks) are based on many parameters. Even in the case that we know their respective weights, their combination does not provide an understandable explanation of the algorithm decision (Burrell, 2016). As a result, these systems also are opaque to their developers.

Lack of transparency or explainability is also known as opacity. It may be due to several reasons (Burrell, 2016). It has been argued that reading the source code of systems embodying intelligent algorithms could be a way of removing such opacity. Unfortunately, these claims underestimate the technical complexity of such systems, as explained above. Obviously, opacity also is intentional from companies in order to have advantage over their competitors and to avoid law regulations. Another technical cause of opacity is that these systems typically do not rely on a single algorithm, but on several algorithms or modules (Jannach et al., 2016).

Opacity has severe implications (Castelluccia & Le Métayer, 2019). Individuals may suffer discrimination, and their privacy and personal data may be threatened. Discrimination may even affect whole collectives, identified by features such as sex or race. Opacity of the inner workings and decisions of intelligent algorithms may produce manipulations and leads to wonder whether some persons or collectives are dealt without dignity and equality of opportunities.

Finally, some implications might involve substantial changes in our society. The emergence of new companies whose business model is based on algorithms may lead to substantial changes in the job market. As a consequence, a debate exists on whether intelligent systems will reduce the number of jobs available or will provide an opportunity for new jobs. Changes also are underway in social habits, from personal relationships by means of social networks to loss of sense of property of certain goods that can be socially shared, e.g., cars or apartments. Hacking artificial-intelligence systems may cause major damage. Recommendations may be biased to extremist points of view, promoting radicalism and challenging respect and tolerance, even in democratic societies. Actually, these systems may be used in ways incompatible with the rules of a democratic society, distorting information and controlling people.

HOW ARE ALGORITHMS UNDERSTOOD IN OTHER DISCIPLINES

The term algorithm is used in other sciences with the two meanings presented above. Not only does science use traditional algorithms for their research, but these sciences have also provided novel problems and challenges to informatics. A paradigmatic example is bioinformatics, which demanded more research on algorithms that involve linear or hierarchical data structures (Gusfield, 1997). Other sciences use informatics to construct computational versions of mathematical models and apply numerical algorithms to solve them, e.g., physics.

References to traditional algorithms can also be found in the social sciences. For instance, the education framework called the Revised Bloom's Taxonomy (Anderson et al., 2001) identifies procedural knowledge as a type of knowledge. A subtype of procedural knowledge is knowledge of subject-specific skills and algorithms. Their description of algorithms is consistent with the description of traditional algorithms given here (Anderson et al., 2001, p. 53): "*a common example is knowledge of algorithms used with mathematical exercises*", such as "*algorithms for solving quadratic equations*". Furthermore, the authors clarify the difference between algorithms and less precise methods by introducing a second subtype of procedural knowledge, namely knowledge of subject-specific techniques and methods. We may read: "*In contrast with specific skills and algorithms that usually end in a fixed result, some procedures do not lead to a single predetermined answer or solution*", such as "*methods for evaluating health concepts*" (Anderson et al., 2001, p. 54).

Algorithms are amenable to analysis in any other field of human knowledge, including the social sciences or philosophy (Goffey, 2008; Hill, 2016). However, these studies must be made on the basis of unambiguously identifying the algorithms addressed (e.g., Burrell, 2016), not on inexact simplifications. Thus, it is not sound to give the definition of traditional algorithms to refer to intelligent algorithms, or to question the mathematical properties of traditional algorithms, by attributing them properties of intelligent algorithms. Unfortunately, many (and probably most) studies on the social impact of intelligent algorithms do not make a distinction between the two types of algorithms. The confusion between algorithm and program, identified above, also is common in social studies. There are exceptions, as Mittelstadt et al. (2016), who explicitly state that they refer to algorithms as understood by the general public, not to "*algorithms that automate mundane tasks*" (p. 3).

Obviously, there are many aspects of traditional algorithms that are interesting for the social sciences, such as algorithm learning and understanding (Velázquez-Iturbide, 2019), the process of either algorithm design (Ginat, 2001) or analysis (De Millo et al., 1979), and even the interests and politics that influence which problems and algorithms are at the mainstream of research. However, this does not contradict the fact that traditional algorithms are abstract objects that can be studied both formally and experimentally, similarly to other mathematical or engineering products (Burrell, 2016).

The academic seriousness proposed here about differentiating different types of algorithms could contribute to solve communication problems among disciplines. This mismatched communication does not only occur between computer scientists and other experts. As Moats and Seaver (2019) found, social science scholars that study algorithms and data scientists face communication challenges with respect to their shared terminology but unshared meanings, i.e., differences in understanding of the world which lead to different operationalization of units of analysis and research goals. Thus, recognizing the ontological differences among disciplines, it is more important than ever to claim that algorithms should be defined clearly, as it has been outlined throughout this chapter. Otherwise, this will only lead to more disagreements which will make it harder for interdisciplinary studies.

FUTURE RESEARCH DIRECTIONS

Research efforts are under way for both traditional and intelligent algorithms. Research on traditional algorithms is especially relevant in specific domains, such as cybersecurity, or from novel approaches, such as meta-heuristics. However, the bulk of research lies on intelligent algorithms. Despite the ill definition of the problems they address, intelligent algorithms are of high interest to companies. Research on them experienced a dramatic increase in past decades and will continue being expanded in the future.

One hot topic of interest is the opacity of intelligent algorithms. Research is under way to combine high performance of novel intelligent systems and the explainability features of symbolic artificial intelligence systems. Achieving a trade-off between the reluctance of companies to law regulations and the right of citizens to have guarantees of fairness is another challenge.

From a technical point of view, it has been suggested that designers should not always attempt to construct intelligent systems capable to give an acceptable outcome in one shot. Alternatively, they could be assistants who could cooperate with the user in a dialog aimed at satisfying his/her information needs (Jannach et al., 2016).

The most recent trend, which has produced a high impact on the public opinion, is generative artificial intelligence. This extreme form of intelligent algorithms is aimed at generating such human products as paintings, music or even computer programs (Kirkpatrick, 2023). Their implications probably will be severe and long-lasting on many fields of knowledge and social sectors. It even is impacting informatics itself (Denning, 2023), where generation of code is forcing to wonder whether informatics education must be revised and even some informatics professions, especially programmers.

CONCLUSION

We have presented a detailed description of the term "algorithm" (within the limits of a book chapter). Thus, we have analyzed both the traditional view of algorithms, as studied in informatics, as well as the novel, intelligent algorithms that are an object of public interest and debate. By presenting their main characteristics and differences, we hope to have contributed to a better understanding, both to computer scientists and to interested readers from other areas of knowledge. Ultimately, we hope that enhanced conceptual understanding may enhance the potential for interdisciplinarity.

ACKNOWLEDGMENT

This work was partially supported by research grants PROGRAMA of Universidad Juan Carlos (ref. M3035) and Ministerio de Ciencia e Innovación (ref. PID2022-137849OB-I00).

REFERENCES

Abiteboul, S., & Dowek, G. (2020). *The age of algorithms*. Cambridge University Press. doi:10.1017/9781108614139

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, R., & Wittrock, M. C. (2001). *A taxonomy for learning, teaching and assessing. A revision of Bloom's taxonomy of educational objectives*. Pearson Education.

Association for Computing Machinery & Computer Society of the Institute of Electrical and Electronics Engineers. (2013). *Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science*. doi:10.1145/2534860

Atchison, W. F., Conte, S. D., Hamblen, J. W., Hull, T. E., Keenan, T. A., Kehl, W. B., McCluskey, E. J., Navarro, S. O., Rheinboldt, W. C., Schweppe, E. J., Viavant, W., & Young, D. M. Jr. (1968). Curriculum 68: Recommendations for academic programs in computer science. *Communications of the ACM*, *11*(3), 151–197. doi:10.1145/362929.362976

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. doi:10.1145/3015455

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132. doi:10.1016/j.knosys.2013.03.012

Böhm, C., & Jacopini, G. (1966). Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, *9*(5), 366–371. doi:10.1145/355592.365646

Boyar, J., Favrholdt, L. M., Kudahl, C., Larsen, K. S., & Mikkelsen, J. W. (2017). Online algorithms with advice: A survey. *ACM Computing Surveys*, *50*(2), 19. Advance online publication. doi:10.1145/3056461

Brassard, G., & Bratley, P. (1996). *Fundamentals of algorithmics*. Prentice-Hall.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems, 30*(1-7), 107-117. doi:10.1016/S0169-7552(98)00110-X

Burrell, J. (2016). How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, *3*(1), 1–12. doi:10.1177/2053951715622512

Castelluccia, C., & Le Métayer, D. (2019). *Understanding algorithmic decision-making: Opportunities and challenges*. doi:10.2861/536131

Computer Science Teachers Association & International Society for Technology in Education. (2011). *Operational definition of computational thinking for K–12 education*. https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

Dan, O., & Davison, B. D. (2016). Measuring and predicting search engine users' satisfaction. *ACM Computing Surveys*, *49*(1), 8. Advance online publication. doi:10.1145/2893486

De Millo, R. A., Lipton, R. J., & Perlis, A. J. (1979). Social processes and proofs of theorems and programs. *Communications of the ACM*, *22*(5), 271–280. doi:10.1145/359104.359106

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. doi:10.1145/2998438

Denning, P. J. (2023). Can generative AI bots be trusted? *Communications of the ACM*, *66*(6), 24–27. doi:10.1145/3592981

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., & Young, P. R. (1989). Computing as a discipline. *Communications of the ACM*, *32*(1), 9–23. doi:10.1145/63238.63239

Denning, P. J., Tedre, M., & Yongpradit, P. (2017). Misconceptions about computer science. *Communications of the ACM*, *60*(3), 31–33. doi:10.1145/3041047

Dijkstra, E. W., & Feijen, W. H. (1988). *A method of programming*. Addison-Wesley.

Edwards, M. (2011). Algorithmic composition: Computational thinking in music. *Communications of the ACM*, *54*(7), 58–67. doi:10.1145/1965724.1965742

Forišek, M., & Steinová, M. (2012). Metaphors and analogies for teaching algorithms. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12*, ACM Press. 10.1145/2157136.2157147

Geiger, R. S. (2017). Beyond opening up the black box: Investigating the role of algorithmic systems in Wikipedian organizational culture. *Big Data & Society*, *4*(2), 1–14. doi:10.1177/2053951717730735

Gillespie, T. (2014). The relevance of algorithms. In T. Gillespie, P. J. Boczkowski, & K. A. Foot (Eds.), *Media technologies: Essays on communication, materiality, and society* (pp. 167–193). The MIT Press.

Ginat, D. (2001). Misleading intuition in algorithmic problem solving. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education, SIGCSE '01*, 21-25. 10.1145/364447.364530

Goffey, A. (2008). Algorithm. In M. Fuller (Ed.), *Software studies – A lexicon* (pp. 15–20). The MIT Press. doi:10.7551/mitpress/7725.003.0004

Gusfield, D. (1997). *Algorithms on strings, trees and sequences: Computer science and computational biology*. Cambridge University Press. doi:10.1017/CBO9780511574931

Hill, R. K. (2016). What an algorithm is. *Philosophy & Technology*, *29*(1), 35–59. doi:10.100713347-014-0184-5

Hoare, C. A. R. (1961). Algorithm 64: Quicksort. *Communications of the ACM*, *4*(7), 321. doi:10.1145/366622.366644

Horowitz, E., Sahni, S., & Rajasekaran, S. (1997). *Computer algorithms*. Computer Science Press.

Jannach, D., Resnick, P., Tuzhilin, P., & Zanker, M. (2016). Recommender systems - Beyond matrix completion. *Communications of the ACM*, *59*(11), 94–102. doi:10.1145/2891406

Kernighan, B. W., & Ritchie, D. M. (1978). *The C programming language*. Prentice Hall.

Kirkpatrick, K. (2023). Can AI demonstrate creativity? *Communications of the ACM*, *66*(2), 21–23. doi:10.1145/3575665

Kleinberg, J., & Tardos, É. (2006). *Algorithm design*. Pearson Addison-Wesley.

Knuth, D. E. (1972). Ancient Babylonian algorithms. *Communications of the ACM*, *15*(7), 671–677. doi:10.1145/361454.361514

Knuth, D. E. (1973). The art of computer programming: Vol. 1. *Fundamental algorithms* (2nd ed.). Addison-Wesley.

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly*, *92*(3), 170–181. doi:10.1080/00029890.1985.11971572

Manna, Z., & Waldinger, R. (1980). A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, *2*(1), 90–121. doi:10.1145/357084.357090

Manouselis, N., Drachsler, H., Verbert, K., & Duval, E. (2013). *Recommender systems for learning*. Springer. doi:10.1007/978-1-4614-4361-2

McGeoch, C. C. (2012). *A guide to experimental algorithmics*. Cambridge University Press. doi:10.1017/CBO9780511843747

Mittelstadt, B. D., Allo, P., Taddeo, M., Watcher, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. *Big Data & Society*, *3*(2). Advance online publication. doi:10.1177/2053951716679679

Moats, D., & Seaver, N. (2019). "You social scientists love mind games": Experimenting in the "divide" between data science and critical algorithm studies. *Big Data & Society*, *6*(1). Advance online publication. doi:10.1177/2053951719833404

Perrenet, J., Groote, J. F., & Kaasenbrood, E. (2005). Exploring students understanding of the concept of algorithm: Levels of abstraction. *Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2005*, 64-68. 10.1145/1067445.1067467

Seaver, N. (2019). Knowing algorithms. In J. Vertesi & D. Ribes (Eds.), *DigitalSTS: A field guide for science & technology studies* (pp. 412–422). Princeton University Press. doi:10.2307/j.ctvc77mp9.30

Velázquez-Iturbide, J. Á. (2019). Students' misconceptions of optimization algorithms. *Proceedings of the 24th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2019*, 464-470. 10.1145/3304221.3319749

Velázquez-Iturbide, J. Á. (2023). Designing exercises for block-based languages: The case of ScratchJr. In F. J. García-Peñalvo & A. García-Holgado (Eds.), *Proceedings TEEM 2022: Tenth International Conference on Technological Ecosystems for Enhancing Multiculturality*. Springer. 10.1007/978-981-99-0942-1_5

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. doi:10.1145/1118178.1118215

Zangerle, E., & Bauer, C. (2022). Evaluating recommender systems: Survey and framework. *ACM Computing Surveys*, *55*(8), 170. Advance online publication. doi:10.1145/3556536

ADDITIONAL READING

Bentley, J. (2000). *Programming pearls* (2nd ed.). Addison-Wesley.

Erkan, A., Barr, J., Clear, T., Izu, C., López del Álamo, C. J., Mohammed, H., & Nadimpalli, M. (2018). Developing a holistic understanding of systems and algorithms through research papers. *Proceedings of the 2017 ITiCSE Conference on Working Group Reports, ITiCSE-WGR '17*, 86-104. 10.1145/3174781.3174786

Haberman, B., Averbuch, H., & Ginat, D. (2005). Is it really an algorithm – The need for explicit discourse. *Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2005*, 74-78. 10.1145/1151954.1067469

Livnat, A., & Papadimitriou, C. (2016). Sex as an algorithm: The theory of evolution under the lens of computation. *Communications of the ACM*, *59*(11), 84–93. doi:10.1145/2934662

MacCormick, J. (2012). *Nine algorithms that changed the future*. Princeton University Press. doi:10.1515/9780691209050

Michalewitz, Z., & Fogel, D. B. (2004). *How to solve it: Modern heuristics* (2nd ed.). Springer Verlag. doi:10.1007/978-3-662-07807-5

O'Neil, C. (2017). *Weapons of math destruction*. Penguin Books.

Pessach, D., & Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys*, *55*(3), 51. Advance online publication. doi:10.1145/3494672

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Pearson Education.

Wirth, N. (1976). *Algorithms + data structures = programs*. Prentice-Hall.

KEY TERMS AND DEFINITIONS

Algorithm: A precise computational procedure that, given valid input data, computes output data that satisfies the problem statement it is intended to solve.

Artificial Intelligence (AI): There are many definitions of artificial intelligence, depending on the area of human knowledge. In informatics, it refers to an area of informatics that attempts to solve problems that usually are considered to be of exclusive competence of human beings.

Computational Problem: A challenge consisting in computing an output value outcome from valid input data, such that a certain relationship is satisfied between the input and output data.

Heuristic: A rule of thumb or an approach to solve a problem that is not guaranteed to be accurate or optimal, but is nevertheless sufficient for reaching an outcome, typically in less time than a more accurate algorithm.

Machine Learning: A field of AI aimed at developing methods that let machines "learn" how to solve ill-defined problems from previously available data.

Opacity: Lack of information about the details of an algorithm or, at least, about the decisions that led to an outcome for a specific query.

Program: A valid sentence in a programming language, constructed according to the rules of the language. If a processor of the language is available, the program can be executed by a computer.

Recommender System: An AI software system that helps a user to find items of interest in a situation of information overload.

Search Engine: A software system that, given a query composed of words, recommends a ranked list of web pages related to the words.

Time Complexity: A function that presents the temporal performance of an algorithm, modeled as the number of basic operations performed by any input data. It is summarized by its asymptotic order, i.e., the term of the function with the highest weight on time performance.

ENDNOTES

1 "Informatics" and "computing" are synonyms, thus they can be used interchangeably. The former is used in continental Europe; the latter is used in English-speaking countries.

2 https://en.wikipedia.org/wiki/Algorithm