Speculative computation - application scenarios

João Ramos *, Tiago Oliveira, Davide Carneiro, Ken Satoh and Paulo Novais

Abstract Artificial Intelligence and Machine Learning have been widely applied in several areas with the twofold goal of improving people's wellbeing and accelerating computational processes. This may be seen in medical assistance (for example, automatic verification of MRI images), in personal assistants that adapt the content to the user based on his/her preferences or, to optimize query response times in relational databases and accelerate the information retrieval process. Most of Machine Learning algorithms used need a dataset to train on, so that the resulting models can be used, for example, to predict a value or enable user-specific results. Considering predictive methods, when new data arrives a new training of the model may be needed. Speculative Computation is a Machine Learning sub-field that seeks to enable computation to be one step ahead of the user by speculating the value that will be received to be computed. A change in the environment may affect the execution, but the adjustments are rapidly performed. This paper intends to provide an overview of the field of Speculative Computation, describing its main characteristics and advantages, and different scenarios of the medical field in which it is applied. It also provides a critical and comparative analysis with other Machine Learning

João Ramos
CIICESI, Escola Superior de Tecnologia e Gestão, Politécnico do Porto, Portugal e-mail: jrmr@estg.ipp.pt

Tiago Oliveira
Algoritmi Centre, University of Minho, Braga, Portugal e-mail: toliveira@di.uminho.pt

Davide Carneiro
CIICESI, Escola Superior de Tecnologia e Gestão, Politécnico do Porto, Portugal

Ken Satoh
National Institute of Informatics, Tokyo, Japan e-mail: ksatoh@nii.co.jp

Paulo Novais
Algoritmi Centre, University of Minho, Braga, Portugal e-mail: pjon@di.uminho.pt

* corresponding author

methods, and a description of how to apply different algorithms to create better systems.

1 Introduction

Digital data are increasing in volume every day. Thus, companies and researchers are seeking new methods to deal with such raw resources in a faster and less resource-dependent way. A major goal is to extract valuable information from raw data in order to, for example, predict a value or to adapt the results to user preferences. In huge volumes of data is frequent the appearance of patterns, which may occur at specific situations, or repeat at a specific cycle time. This pattern evaluation is time-consuming and may be impossible to detect to the human-being. Though the study of data with date labels (denoted as time-series) is possible to detect trends as, for example, sales trends and take better selling decisions. Adeyemi et al. (2018) studied the data flow in an University for a 12-months period and created a set of statistics and graphics which allowed to identify possible trends. These may be especially important to ensure a better quality of the service. Similarly, in Adekitan et al. (2019) the authors went as far as developing a model to predict Internet traffic.

Deep Learning (DL) approaches for computer vision (Voulodimos et al. 2018), for example, enable the machine to understand the objects that are present in a figure. These approaches also have other application scenarios such as human activity recognition (Ronao and Cho 2016). In classification tasks, other Machine Learning (ML) methods such as Support Vector Machines (SVM) or Linear Regression (LR) can be used. The aforementioned methods are able to predict a value using a model that is built based on a previous training phase. The use of such algorithms and models implies a present-oriented and real-time target, *i.e.*, they are not be able to anticipate future value since the future input data is not yet available.

Other techniques, such as those deemed as unsupervised learning, may not require a training step. However, the quality of the results obtained may not be assured since there is no ground truth to compare with. Nevertheless, these techniques are especially used in anomaly detection or collaborative filtering (for example, to estimate user preferences).

Speculative Computation represents an entirely different take on the problem. It is a technique that is able to anticipate the possible answer. Considering, for example, Neural Networks, it will estimate a given input and, instead of being in a idle state while the actual input is not received, it tries to speculate its possible value. Thus, the computation continues and, if necessary, when the actual value is received, the computation is revised. This kind of techniques have, however, been used in other domains, such as in spectre attacks Kocher et al. (2020).

The main objective of this paper is to analyze the advantages and disadvantages of applying speculative computation, through a comparison with other Machine Learning (ML) techniques whenever possible. Two applications of this type of computation are also analyzed and demonstrated.

The paper is divided into five sections, which are organized as follows: Section 2 contains the main supervised and unsupervised machine learning techniques; Section 3 details the Speculative Computation method and, in Section 4 two different cases of application are described. Finally, in Section 5, the main conclusions are drawn and future investigation lines are highlighted.

2 Literature Review: ML techniques

Machine Learning is considered a branch of Artificial Intelligence (AI). In this field, the goal is to create mechanisms to mimic the human behaviour in what concerns learning, reasoning and problem solving and, eventually, achieve a performance that surpasses human capabilities. For this kind of processes, there is the need for large sets of data in order to provide the learning algorithms with a representative range of the different possibilities. The algorithms that fall under this field use statistical methods in order to be trained and to make classifications or predictions based on trained data. The use of big data enables the algorithms to uncover key insights and to better assist the decision maker.

Artificial Intelligence and ML are often used interchangeably, but AI has a broader goal, *i.e.*, AI intends to enable computers to mimic human problem-solving and decision-making abilities. Instead of providing a prediction or a classification and enabling the human to take an action, AI goes further and also perform this last step. Most common examples of real world applications of AI systems are speech recognition, computer vision, text translation or recommendation systems.

Focusing on ML algorithms, there are several prominent methods. The main goal of the paper is not to provide a detailed description of each method, but to identify the ones that may be used and compared to speculative computation.

In broad terms, ML algorithms can be categorized as supervised or unsupervised. The former makes use of labeled data to train a model, which is then able to classify or predict outcomes for new unlabeled data. Overfitting and underfitting are common problems to supervised learning algorithms, depending on the characteristics of the input data. On the other hand, unsupervised learning is able to find patterns in unlabeled data. The goal is to discover hidden patters or to group data that may be considered similar. Table 1 summarizes the most prominent algorithms in Machine Learning.

Table 1 Most prominent Machine Learning algorithms.

Supervised Learning	Unsupervised Learning
Neural Networks	k-means clustering
Convolutional Neural Networks	Principal Component Analysis
Linear/Logistic Regression	
Random Forest	
Support Vector Machines	

Neural Networks were inspired by biological systems and are able to model complex problems (Vadla et al. 2021). These networks are composed by multiple interconnected units called neurons and the connections represent synapses. Neurons are placed on a set of layers where the first and last are deemed the input and output layer, respectively. Intermediary layers, also named hidden layers, may have different activation functions. The goal of the activation function is to simulate the process that goes on in the mammal's nervous system, in which a neuron's input value must be above a given threshold to activate that neuron and send information to the next layer.

A special type of Neural Networks is the Convolutional Neural Network (CNN) (Khan et al. 2020). It diverges from traditional NN due to the existence of convolutional layers, which provide powerful learning ability. Indeed, there are multiple stages that provide feature extraction and enable the CNN to automatically learn different representations of the data. CNNs have successfully been applied on Computer Vision and Image Processing, Natural Language Processing and Speech Recognition (Khan et al. 2020; Voulodimos et al. 2018). Contrary to a NN that have a set of interconnected layers of neurons, a CNN is composed by a set of convolutional layers, pooling layers and fully-connected layers.

Logistic Regression is applied to classify a given input, usually a nominal variable, as belonging or not to a given group, *i.e.*, output values may be of yes-no type, inclusion or non-inclusion (George and Mallery 2018). Linear or multiple regression has the goal to output a continuous variable (predict a value) considering one or more independent variables. Pun et al. (2019) used a multiple regression to estimate traffic flow in a city. In their study, the authors needed to analyze the correlation between independent variables and the dependent variable and defined which factors contributed to estimate the traffic flow accurately.

Contrary to linear regression, that assumes that there is linearity between the variables, the Random Forest algorithm does not make such assumption. Thus, this algorithm is more flexible, being able to better adapt to nonlinearities and, therefore, to make better predictions for such type of data (Schonlau and Zou 2020). This statistical learning algorithm is often used on datasets where the number of observations is smaller than the number of variables. In this case, linear or logistic regression may not be used. Random Forests may be applied on classification and regression problems alike.

Support Vector Machine (SVM) is typically used for classification and the decision function of the algorithm is an optimal hyperplane that separates the dataset with the best accuracy. Thus, SVM tries to indicate the class that an observation belongs to based on patterns of information (Pisner and Schnyer 2020).

When facing unlabeled data, the previously described algorithms cannot be applied since the lack of a label renders the learning/fitting task, in which a cost function is minimized, impossible. In the absence of labeled data, the algorithms that are able to extract insights are those under the unsupervised learning umbrella.

K-means (as depicted in Table 1) is an algorithm that tries to create groups of data, *i.e.*, clusters. Each element present in a cluster is consider to be similar to others in the same cluster, but different from those of other clusters. Indeed, k-means

clustering is a method for partitioning data objects according to perceived intrinsic characteristics (Xie et al. 2019). This algorithm is applied in image segmentation, text mining, among other problems. The creation of clusters may be performed by partitioning or by hierarchical methods. The former creates k clusters simultaneously and each data sample is attributed to one cluster. The latter creates a hierarchy of clusters, in which a top-level cluster may be subdivided into two or more clusters. Despite some limitations, such as noise susceptibility, k-means clustering is an efficient algorithm and is easy to implement (Xie et al. 2019).

Principal Component Analysis (PCA) may be considered as one of the most important unsupervised techniques in data mining (Gewers et al. 2021). PCA may be briefly described as a statistical method that applies a rotation to the axes of the feature vector. This rotation, among each axis of the vector, considers the direction of maximum possible data dispersion. Given the results of such rotation, and that the axes are ordered by its maximum data dispersion (*i.e.*, the first axis has the highest value), the axes with the lowest dispersions may be discarded without great loss of data variation. Thus, reducing the number of axes is equal to removing the size of the feature space and enables the application of other ML techniques in a more efficient way. Indeed, the original data (which contains a high vector space) is simplified with minimal loss, thus reducing its dimensionality (Gewers et al. 2021). This technique is applied, for example, for data visualization when the number of features (number of vector axis) is higher than three and plots are hard to create and interpret.

Other types of learning exist, such as Semi-supervised Learning or Reinforcement Learning (RL), although their use is less frequent. In Semi-supervised Learning, labeled and unlabeled data are used together. Thus, these methods fall somewhat between Supervised and Unsupervised Learning. Reinforcement Learning, on the other hand, is similar to supervised learning, but the training of the algorithm is not supported by training examples. Instead, using a reward function and interacting with the world, the algorithm is able to self-train through trial and error. When an action provides a good result, the reward obtained is high in contrast with an undesirable result. Thus, successful outcomes reinforce the associated behaviors, while unsuccessful one weaken them.

3 Speculative Computation

The anticipation ability may be considered an essential feature for a reasoning system and, more specifically, for the machine learning process. Indeed, by starting the learning job as early as possible, it is possible to accomplish it sooner. A goal of Speculative Computation is to start part of the work before being necessary (Burton 1985). As stated in (Burton 1985) “if work which is known to be necessary (mandatory work) is given priority over other work (speculative work), then performing speculative work can only speed computation”. Thus, computing such non-essential jobs in parallel, jobs that are speculated to be useful in the near future, will accel-

erate computation and achieve results in less time. The past work of (Burton 1985) was a simple functional feature to control the speculative work. More recently, other frameworks have been proposed that explore this main feature of speculative computation, such as Satoh et al. (2000); Satoh and Yamamoto (2002); Satoh (2005); Hosobe et al. (2007); Fukuta et al. (2008); Boudol and Petri (2010).

Satoh et al. (2000) developed a method for solving the problem of communication in a multiagent system. In their work, the authors applied speculative computation to anticipate the answer of an agent to another. By using communication channels such the Internet, the messages may be delayed (or lost), but agents' processes should not be stopped. For this process, the developed framework makes use of three objects: current computation status; already used defaults (values used when the real information is missing or has not been received); and answers substitutions for the initial query (received values that may differ from the default ones used). The framework was extended to a master-slave multiagent system (Satoh and Yamamoto 2002), where the authors applied Speculative Computation to the task of booking an hotel room, with two personal agents (representing two persons) and the hotel room reservation agent (that acted as the hotel receptionist).

Satoh, in (Satoh 2005), defined the master-slave Speculative Computation system framework with the following tuple $\langle \Sigma, \mathscr{E}, \Delta, \mathscr{A}, \mathscr{P}, \mathscr{I} \rangle$. The elements of the tuple are defined as:

- Σ is a finite set of constants and each element is denoted as a system module;
- $\mathscr{E}$ is the set of *external predicates*, where each elements is a function. Considering Q as a literal that belongs to a given external predicate and S the information source identifier, $Q@S$ is an *askable literal*. The negative form of the askable literal is defined as $\sim(Q@S)$ or $(\sim Q)@S$;
- Δ denotes the *default answer* set, which contains the ground askable literals. For this set the following askable literals may not be simultaneously present
  - ▷ $p(t_1,\ldots,t_n)@S$; and
  - ▷ $\sim\ p(t_1,\ldots,t_n)@S$
- $\mathscr{A}$ corresponds to the set of *abducible predicates*. The literal Q is *abducible* whenever it is a literal with an *abducible predicate*;
- $\mathscr{P}$ is the set of rules of the Logic Program (LP). Such rules are in the form:
  - ▷ $p \leftarrow p_1, p_2, \ldots, p_n$, where
    - · p is a positive ordinary literal, and
    - · each of $p_1,\ldots,p_n$ may be an ordinary literal, an askable literal or an abducible;
  - ▷ the head of rule R is represented by p, which may also be written as as $head(P)$. The rule's head must always be non-empty. The body of the rule R, $body(P)$, is represented by $p \leftarrow p_1,\ldots,p_n$. For some specific situations the body may be replaced by the boolean value *true*.
- $\mathscr{I}$ denotes the set of integrity constraints in the form:

$$?(p_1, p_2, \ldots, p_n)$$

- ▷ the symbol "?" represents "falsity", *i.e.* contradiction,
- ▷ p_1, p_2, …, p_n may be ordinary literals, *askable literals* or *abducibles*
- ▷ at least one of $p_1, p_2, \ldots, p_n$ is an *askable literal* or an *abducible*.

According to the set that an *askable literal* be, there is a different meaning. Indeed, an *askable literal* may be:

1. a question put to a system module S, when $Q@S$ is in a rule $\mathscr{P}$; or
2. a default truth value (*true* or *false*), when it is present in Δ

- $p(t_1,\ldots,t_n)@S \in \Delta$, $p(t_1,\ldots,t_n)@S$ is usually *true* for a question to a system module S, and
- $\sim p(t_1,\ldots,t_n)@S \in \Delta$, $p(t_1,\ldots,t_n)@S$ is generally *false* for a question to a system module S.

Considering the aforementioned definitions, the execution of the Speculative Computation framework enables the process to continue even under an incomplete information scenario. Indeed, when some data are needed, an information source is queried through an askable literal, and the computation continues by using the default value. Whenever new data are retrieved by an information source, the computation is revised and, if necessary, new execution branches are started. Speculative Computation may be considered an iterative process that at one point is in the *Process Reduction Phase* in order to obtain a tentative solution and, when a real value is obtained, changes to the *Fact Arrival Phase* (Fig. 1).

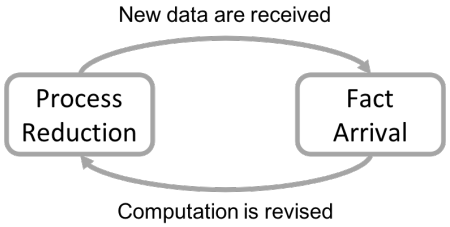

Fig. 1 Speculative computation cycle

Formal definitions of the processes that are executed in each phase are given in Sections 3.1 through 3.4.

3.1 Preliminary Definitions

In order to apply the Speculative Computation framework and better understand its two phases (described in Sections 3.4 and 3.3) there are important concepts that must be defined.

Definition 1 *An extended literal is either a literal or an expression of the form* $fail(\{l_1,\ldots,l_n\})$ *where* l_i *is a literal (Satoh 2005).* ■

The keyword *fail* is used to prove that there is no valid proof for l_i.

Definition 2 *The tuple* $\langle GS, OD, IA, ANS\rangle$ *defines a process where*

- *GS (Goal Set) denotes the set of extended literals. This set also defines the current computation status of an alternative solution;*
- *OD (Outside Defaults) denotes the set of askable literals, which is the set that contains the assumed information regarding the outside world;*
- *IA (Inside Assumptions) represents the set of negative literals or abducibles, considering the values that were assumed during a computation; and*
- *ANS (Answers) denotes the set with the initials values (instantiated values) for the variables. These values are used to start the computation.* ■

Definition 3 *The Process Set (PS) contains a subset of askable literals denoted Already Asked Questions (AAQ), and a subset of askable literals called Current Belief State (CBS).* ■

The *Process Set* contains all execution branches (alternative computations) that have been considered at a given computation point. During the execution, the *AAQ* reduces the number of duplicated questions to the information sources. Indeed, before questioning a source for a given value, the computation verifies if such question has been posed to the information sources and only if the question is asked for the first time, the information source is queried. The *Current Belief State (CBS)* denotes the current status of the outside world. During the execution, processes may be in an *active* or *suspended* state, exchanging it state according to the current computation status and the information retrieved by the information sources.

Definition 4 *Let* $\langle GS, OD, IA, ANS\rangle$ *be a process and CBS be a current belief state. A process considered to be active with respect to CBS if* $OD \subseteq CBS$*. A process is considered as suspended with respect to CBS otherwise.* ■

The definition of an active process emphasizes that it is a process whose outside defaults have to be consistent with the current belief state.

3.2 Process Reduction Phase

The Process Reduction Phase represents the active execution of the computation, *i.e.*, the process that queries the information source for real world data and contin-

ues the execution using the default assumed values until such information is obtained. Indeed, the Process Set may be changed during this execution. In order to demonstrate the changes in the process set, changed *PS*, *AAQ* and *CBS* are defined as *NewPS*, *NewAAQ* and *NewCBS*; otherwise they stay unchanged.

Initial Step: Let GS be an initial goal set. The tuple $\langle GS, \emptyset, \emptyset, ANS \rangle$ is given to the proof procedure where ANS is a set of variables in GS. That is, $PS = \{\langle GS, \emptyset, \emptyset, ANS \rangle\}$. Let $AAQ = \emptyset$ and $CBS = \Delta$.

Iteration Step: Do the following:

- ▷ **Case 1:** If there is an active process $\langle GS, \emptyset, \emptyset, ANS \rangle$ with respect to CBS in PS, terminate the process by returning outside defaults OD, inside assumptions IA, and instantiation for variables ANS. This case may only be applied on the first iteration step since OD and IA are empty sets;
- ▷ **Case 2:** If there is no active process, terminate the process by reporting a failure of the goal;
- ▷ **Case 3:** Select an active process $\langle GS, OD, IA, ANS \rangle$ with respect to CBS from PS and select an extended literal L in GS. Let $PS' = PS - \{\langle GS, OD, IA, ANS \rangle\}$ and $GS' = GS - \{L\}$. For the selected extended literal L, do the following:
  - **Case 3.1:** If L is a positive ordinary literal, $NewPS = PS' \bigcup \{\langle (\{ body(R)\} \bigcup GS')\theta, OD, IA, ANS\theta \rangle | \exists R \in \mathscr{P}$ and $\exists$most general unifier θ so that $head(R)\theta = L\theta$.
  - **Case 3.2:** If L is a ground negative ordinary literal or a ground abducible then:
    - **Case 3.2.1:** If $L \in IA$ then $NewPS = PS' \bigcup \{\langle GS', OD, IA, ANS \rangle\}$.
    - **Case 3.2.2:** If $\overline{L} \in IA$ then $NewPS = PS'$.
    - **Case 3.2.3:** If $L \notin IA$ then $NewPS = PS' \bigcup \{\langle NewGS, OD, IA, \bigcup\{L\}, ANS \rangle\}$ where $NewGS = \{fail(BS) | BS \in resolvent(\ L, \mathscr{P} \bigcup \mathscr{I})\} \bigcup GS'$ and $resolvent(L,T)$ is defined as follows:
      - If L is a ground negative ordinary literal, $resolvent(L,T) = \{\{L_1\theta, \ldots, L_k\theta\} | H \leftarrow L_1, \ldots, L_k \in T$ so that $\overline{L} = H\theta$ by a ground substitution $\theta\}$
      - If L is a ground abducible, $resolvent(L,T) = \{\{L_1\theta, \ldots, L_{i-1}\theta, L_{i+1}\theta, \ldots, L_k\theta\} | \bot \leftarrow L_1, \ldots, L_k \in T$ so that $L = L_i\theta$ by a ground substitution $\theta\}$.
  - **Case 3.3:** If L is $fail(BS)$, then
    - If $BS = \emptyset$, $NewPS = PS'$;
    - IF $BS \neq \emptyset$, then do the following:
      (1) Select B from BS and let $BS' = BS - \{B\}$.
      (2) **Case 3.3.1:** If B is a positive ordinary literal, $NewPS = PS' \bigcup \{\langle NewGS \bigcup GS', OD, IA, ANS \rangle\}$ where $NewGS = \{fail((\{body(R)\} \bigcup BS')\theta | \exists R \in \mathscr{P}$ and $\exists$MGU θ so that $head(R)\theta = B\theta\}$
      Case 3.3.2: If B is a ground negative ordinary literal or a ground askable literal or an abducible,

$NewPS = PS' \bigcup \{\langle \{ \ fail(BS')\} \bigcup \ GS', OD, IA, ANS\rangle\} \bigcup \{\langle \{\overline{B}\} \bigcup GS', OD, IA, ANS\rangle\}$.

– **Case 3.4:** If L is a ground askable literal, $Q@S$, then do the following:
(1)If $L \notin AAQ$ and $\overline{L} \notin AAQ$, then send the question Q to the slave agent S and $NewAAQ = AAQ \bigcup \{L\}$.

(2)If $\overline{L} \in OD$ then $NewPS = PS'$ else $NewPS = PS' \bigcup \{\langle GS', OD \bigcup \{L\}, IA, ANS\rangle\}$.

3.3 Fact Arrival Phase

The *Fact Arrival Phase* represents an interruption on the normal execution of the computation. Indeed, whenever an information source (sensor) returns a real value, the normal execution of the computation is stopped and revised. Supposing that an answer Q is returned by the information source S regarding the *askable literal* $Q@S$.

Let $L = Q@S$. After finishing the active step of the process reduction, do the following

- If $\overline{L} \in CBS$, then $NewCBS = CBS - \{\overline{L}\} \bigcup \{L\}$
- Else if $L \notin CBS$, then $NewCBS = CBS \bigcup \{L\}$.

After executing the previous step, some *askable literals* that were considered in the initial belief state may be removed. In this case, processes that were being executed considering such *askable literals* are suspended.

3.4 Correctness of the Proof Procedure

The correctness of the proof procedure, initially introduced by Kakas and Mancarella (1990), ensures that the model is stable concerning its semantics. Through this procedure, and applying the concept of integrity constraints, it is ensured that when a literal is positive it is not possible to keep its negative value in the default set.

Definition 5 *Considering T the set of rules and integrity constraints. $\prod_T$ is the set of ground rules that are obtained by the replacement of every variables in all rules or integrity constraint T by each ground term.* ■

Definition 6 *Let T be a set of rules and integrity constraints. Let M be a set of ground atoms and $\prod_T^M$ be the following program: $\prod_T^M = \{H \leftarrow B_1, \ldots, B_l | H \leftarrow B_1, \ldots, B_l, \sim A_1, \ldots, \sim A_h. \in \prod_P$ and $A_i \notin M$ for each $i = 1, \ldots, h.\}$. Let $min(\prod_T^M)$ be the least model of $\prod_T^M$. A stable model for a logic program T is M iff $M = min(\prod_M^T)$ and $\perp \notin M$.* ■

Definition 7 *Consider Θ the set of ground abducibles.* ■

To a process evaluation strategy, when the proof procedure returns an answer with a set of outside defaults and a set of inside assumptions, this answer is correct with respect to a stable model, regarding the inside assumptions of the program.

Theorem 1. *Let LP be the Logic Program that is used in the speculative framework and has a set of integrity constraints that are satisfiable. Consider* $LP = \langle \Sigma, \mathscr{E}, \Delta, \mathscr{A}, \mathscr{P}, \mathscr{I} \rangle$*, where* $\mathscr{P}$ *is a call-consistent logic program. Let GS be reduced to an* $\emptyset$ *when OD is outside defaults. Let IA be the set of inside assumptions, ANS the variables instantiations' set of GS, and CBS the current belief state. Consider* GS' *as a new set that is obtained from GS where all variables in GS are replaced by ANS. Then, there is a generalized stable model* $M(\Theta)$ *for* $\mathscr{P} \bigcup \mathscr{I} \bigcup \mathscr{F}(CBS)$*, such that* $M(\Theta) \models GS'$ *and* $OC \subseteq CBS$ *and* $IA \subseteq \Theta$*.* ■

Proposition 1. *Since* $\mathscr{P}$ *is a call consistent logic program, so is* $\mathscr{P} \bigcup \mathscr{I} \bigcup \mathscr{F}(CBS)$*. Then, an abducible derivation may be constructed (Kakas and Mancarella 1990) using a set of reduction steps applied to* $\langle GS, \emptyset, \emptyset, ANS' \rangle$ *to* $\langle \emptyset, OD, IA, ANS \rangle$ *where* ANS' *is a set of variables in GS. This derivation is correct for generalized stable model semantics for a call-consistent logic program (Kakas and Mancarella 1990). Thus,* $M(\Theta) \models GS\theta$ *and* $OD \subseteq \Theta$*.* ■

For the reduction process, the following strategy is used:

- ▷ Whenever a positive literal is reduced, new processes are created considering the rule order defined in the program, which are unifiable with the positive literal;
- ▷ The left-most literal is always selected, which may be a newly created or newly resumed process.

4 Speculative Computation: Application Scenarios

The aforementioned Speculative Computation framework (Section 3) has an anticipation feature that may not be seen in other reasoning methods. The goal of the current section is to demonstrate two different scenarios in which this type of computation has been applied. In both cases, Speculative Computation is not used to replace a specific reasoning method, but to enhance the abilities of the systems that integrate it.

4.1 CogHelper

CogHelper (Ramos et al. 2018, 2017a,b) is a system which contains different modules to achieve several objectives. The main objective is to assist an impaired person when traveling outside. In this assistance, the indicated path is adjusted to the user

and the Speculative Computation framework is able to anticipate possible user mistakes and alert the user in advance. A different goal is to provide a localization module, which enables caregivers to receive, in real time, the geographic coordinates of the person with cognitive disabilities.

To better understand the target population of the system, it is important to understand the definition of cognitive disability and its possible implications to the daily life of the person with disabilities. According to American Psychiatric Association (2013), a cognitive disability may be defined as a medical condition that involves a person that presents more difficulties in one or more types of tasks when compared to a neurotypic. Such tasks fall under mental effort tasks, like ensure own-safety, correct development of daily work, keep health conditions, tasks that require orientation abilities, among others.

The diagnostic of such condition may not be an easy process since there are several areas to analyze and the degree of incidence may vary. A stroke, for example, may affect different brain areas and the disabilities that may arise may not be easy to classify. Thus, the main criterion to evaluate the incidence degree is the intellectual function. A person with cognitive disabilities usually has more difficulties in accomplishing some specific mental tasks (Schalock et al. 2010).

According to the incidence level, disabilities may be divided into mild, moderate, severe or extreme. For the last two, the individual is usually completely dependent on a caregiver, whereas moderate cognitive disabilities may only imply increased surveillance by the caregiver. The freedom of the person with disabilities is often reduced since she/he may not be able to go outside the home unaccompanied. Indeed, the person with care needs may become a prisoner of her/his own home, and dependent on the availability of the caregiver.

Lack of orientation may be another relevant factor contributing to decreased autonomy. Consequently, there is a social pressure to develop applications or systems that try to mitigate the loss of autonomy and provide the person with disabilities with as much of an ordinary life as possible. Nevertheless, such applications must consider the end-user characteristics into its design in order to maximize its usability and accessibility. By reducing cognitive processing requirements to a minimum, developed applications may be used by a larger number of individuals. For instance, written text may be replaced by pictures, animations and/or sounds (Lanyi and Brown 2010).

CogHelper (Ramos et al. 2018, 2017a,b) is a system that is being developed by the authors that takes into consideration the aforementioned specifities of the end-users. Indeed, the system considers people with cognitive disabilities as the main target, but also takes into consideration the corresponding caregivers. The system is made by three main modules (Fig. 2).

The module dedicated to the person with disabilities consists in a application for smartphone with the main goal of guiding the user to a pre-selected destination. The interface is based on augmented reality, which tried to minimize the mental effort the user need to perform to understand the provided information. Through it, the user is able to orientate the mobile device to any direction and, when the path that should be traveled is line with the device, an arrow overlaps the real image captured

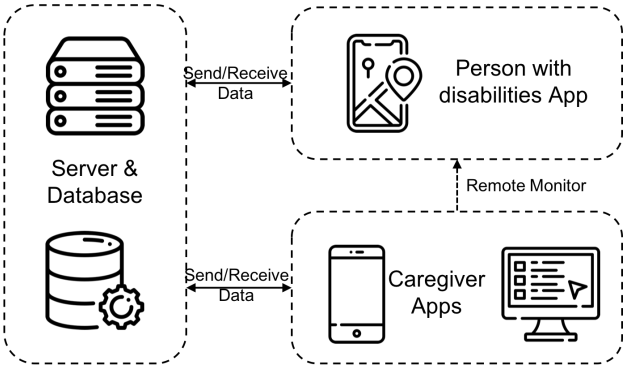

Fig. 2 Coghelper system overview

by the camera. Through the Speculative Computation unit the system is able to anticipate an user mistake and trigger an alert to the user before he/she takes the wrong turn. Considering user preferences and specifities, the indicated travel path is adapted to her/him through a trajectory data mining unit. Here, the Speculative Computation framework is used in symbiosis with the trajectory mining module. Indeed, the default values set of the framework is created based on the selected destination and most frequent streets traveled by the user. However, as depicted in Fig. 3, when there is no historical data of the user, the shortest path is calculated.

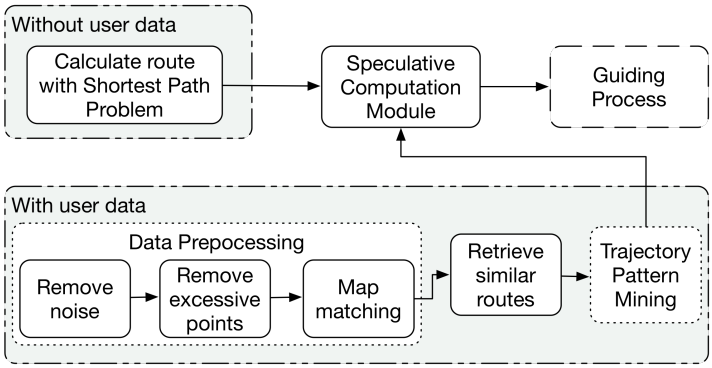

Fig. 3 Pattern mining process, retrieved from Ramos et al. (2018)

When the system has historic data about the user, *i.e.*, when it has some previously traveled paths, it may use them to adapt future paths. Indeed, a data preprocessing module is applied to the paths stored in the database that addresses several situations: to remove points that were mistakenly retrieved by the GPS sensor of the mobile device (for example, a point that is at a bigger distance of the last known location due to signal reflection); to remove excessive points (e.g. remove

unnecessary intermediate locations, such as points between two intersections that only have one possible path). Then, a map matching process is executed in order to reduce raw data variation from the data recorded in geographic information systems (such OpenStreetMap[2]). An example of the result of the map matching module is represented in Fig. 4.

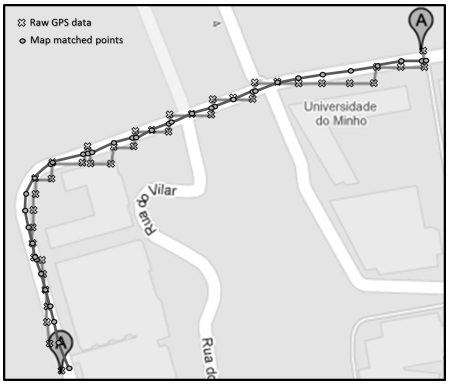

Fig. 4 Trajectory map matching processing, obtained from Ramos et al. (2017b)

After ending the pre-processing stage, similar routes are retrieved by the system, considering the user's current location and selected destination. Then, the trajectory mining module is able to define the entire route to travel or sub-routes (frequent traveled paths) that will be used to calculate the entire travel path. This is then considered as the default set in the Speculative Computation framework, with information regarding locations where the user is systematically taking the wrong turn.

The speculative computation module is an essential piece of the CogHelper system since it provides the anticipation feature. This is important since by alerting the user in advance, the mistake may be avoided and, consequently, the user does not enter in distress. Indeed, before executing the framework, the system requires the following information:

1. The knowledge base has the facts that allow the correct movement between the origin and user desired destination. These facts are all possible paths that enable the user to reach his/her destination;
2. The itineration that is usually executed by the user in the form of default values;
3. The inlcusion of a specific point in the set of recommended points as default values; and
4. The set of rules that assist the computation by considering the path that is more likely to be performed by the user. These rules also include the eventual alert triggering when a deviation from the initial defined path is detected.

[2] `https://www.openstreetmap.org/`

To ensure the correct execution of the speculative computation framework and that the user stays in the correct path, the following structure was specified in terms of the logic programming suite:

- $\Sigma = \{gps\_sensor, recognizer\}$
- $\mathscr{E} = \{user\_travel, included\}$
- $\Delta = \{user\_travel(1,2)@gps\_sensor,$
  $user\_travel(2,3)@gps\_sensor,$
  $user\_travel(3,4)@gps\_sensor,$
  $user\_travel(3,6)@gps\_sensor,$
  $user\_travel(6,7)@gps\_sensor,$
  $user\_travel(7,8)@gps\_sensor,$
  $included(1)@recognizer,$
  $included(2)@recognizer,$
  $included(3)@recognizer,$
  $included(4)@recognizer,$
  $\sim included(5)@recognizer,$
  $\sim included(6)@recognizer,$
  $included(7)@recognizer,$
  $included(8)@recognizer\}$
- $\mathscr{A} = \{show\_next\_point, show\_user\_warning\}$
- $\mathscr{P}$ is the following set of rules:

$$guide(A,A) \leftarrow .$$
$$guide(A,B) \leftarrow path(A,F), show\_next\_point(F), user\_travel(A,F)@gps\_sensor, guide(F,B).$$
$$guide(A,B) \leftarrow path(A,F), user\_travel(A,F)@gps\_sensor, show\_user\_warning(F), guide(F,B).$$
$$path(1,2) \leftarrow .$$
$$path(2,3) \leftarrow .$$
$$path(2,5) \leftarrow .$$
$$path(3,4) \leftarrow .$$
$$path(3,6) \leftarrow .$$
$$path(5,6) \leftarrow .$$
$$path(6,7) \leftarrow .$$
$$path(7,8) \leftarrow .$$
$$path(8,4) \leftarrow .$$

- $\mathscr{I}$ denotes the following set of integrity constraints or invariants:

$$?(show\_next\_point(F), \sim included(F)@recognizer).$$

$$?(show\_user\_warning(F),\\ included(F)@recognizer).$$

In the previous logic programming *path*(*a*,*b*) denotes a connection between location *a* and *b*. The location that the user should travel to is indicated by *show_next_point*. An alert should be triggered and the user must be advised to take the right turn with *show_user_warning*. Default values for the user path (calculated by the trajectory mining module) are set in Δ. Finally, *user_travel*(*a*,*b*) indicates that the system assumes the correct user displacement from *a* to *b*; and through included(*a*) is possible to indicate that a point *a* is part of the path. The logic programming is complete according to the example provided in Fig. 5.

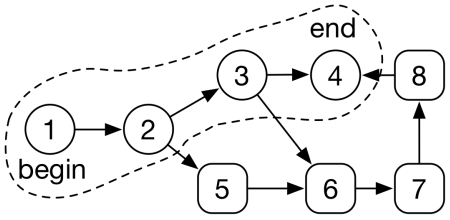

Fig. 5 Tentative (example) paths to travel from 1 to 4

The integrity set of the logic program has two invariants that ensure that the next place shown to the user may not be indicated if it is not part of the path; and that an alert should not be shown to the user if he is traveling through the correct path.

4.2 CompGuide

In the medical field, errors may happen whose consequences may not be reversible. Indeed, during her/his practice, physicians constantly face first-time scenarios where decisions must be taken in short time. Medical errors may occur, which increase spending and loss of quality life for both patients and physicians. To prevent such situations, defensive medicine approaches are adopted, which consist of avoiding difficult cases (and inherent difficult decisions) in order to prevent lawsuits or, ordering a large amount of complementary exams before taking a final decision.

Clinical Practice Guidelines (CPG) were created as a means to disseminate evidence-based medicine. Through such guidelines, physicians may support their decisions based on accepted and approved medical decisions (therapies). CPG's are, as defined by the Institute of Medicine of the United States, systematically developed statements with recommendations about medical procedures considering the clinical patient circumstances (Field and Lohr 1992). CPG's are accepted by healthcare professionals as they are viewed as a means to integrate the most current evidence into the patient management. However, they present some disadvantages

that should be considered: they are very long documents and only a small part of them are actual clinical recommendations. Indeed, the difficulty in consulting such documents are evidenced by their ambiguity regarding misunderstanding of medical terms, conflicting instructions, and incorrect structure of the statements. There are also other types of vagueness such as using probabilistic terms such as *probable* or *unlikely*, making it hard for the physician to quantify and to fit the patient to a specific guideline.

Oliveira et al. (2013) addressed this problem by creating a representation model for CPG's in OWL (Ontology Web Language). Their work was able to take into consideration guidelines of any category (diagnosis, evaluation, management and treatment) and medical specialty (e.g. pediatrics, cardiology). Through this work, Oliveira et al. improved existing systems since they included the definition about clinical constraints, temporal properties, and clinical task scheduling. This enabled the transition from the paper to Computer-Interpretable Guidelines, that benefit from the CPG's formalisms, but are easier to use.

Clinical Practice Guidelines, if correctly understood and used, have the advantage of supporting clinical decisions. Indeed, the guidelines provide an execution careflow with an appropriate order between procedures and the modeling of decision points. At such points, one must choose between alternative paths based on the patient's conditions. Clinical Decision Support Systems consider all information necessary to assist the decision, but whenever an alternative point is reached there may be some missing information and it is impossible to achieve an outcome, since the system waits for the information that may never come. Considering this issue, Oliveira et al. (2014) used Speculative Computation to increase the efficiency of the process by enabling the system to continue the computation of a possible solution while waiting for the necessary information.

Indeed, based on a set of default values, whenever a decision must be taken, the system asks for the real data, but keeps the computation based on the default value. When the data is retrieved from the information source, the computation is revised. In this process, if the default value used is consistent with the returned value, the system is in an advanced stage of computation and there is no delay regarding the provided decision. In the case in which the value is not consistent, the executed branch is paused and a new one is started in order to consider the new value. Oliveira et al. (2014) applied the speculative computation framework to the *CompGuide* model, which enabled the creation of Computer-Interpretable Guidelines.

With this, authors achieved a Clinical Decision Support System that was tested against colon cancer data of several patients. When a patient is diagnosed with this form of cancer, there are a batch of tests that must be performed and the results may take some time. Through the speculative computation framework, the clinical decision support system maintains its execution since there is a possible scenario for each next procedure of the guideline. Thus, a possible clinical decision may be taken (or may be considered) based on the default values (most likely values).

To achieve better results, a next step was taken by Oliveira et al. (2017a) where the default set was created and revised based on Bayesian Networks. Instead of

having a static set of defaults, which are assumed while the real value is not received, this set is dynamically calculated. Indeed, when facts arrive, default values are calculated based on the Bayesian Network, indicating the most likely values. The developed architecture is represented in Fig. 6, which is composed of four main modules:

- guideline engine: interprets the instructions of the clinical guideline based on the current patient state;
- knowledge base: contains the set of guidelines in the form of computer-interpretable guidelines, *i.e.*, a version that the computer is able to use for computation, which is represented in an ontology;
- local repository: contains information about other patients regarding previous executions of the clinical guidelines;
- speculative module: this module is hosted by the guideline engine and implements a speculative framework with dynamically generated default constraints, ensuring the continuous execution of the computation.

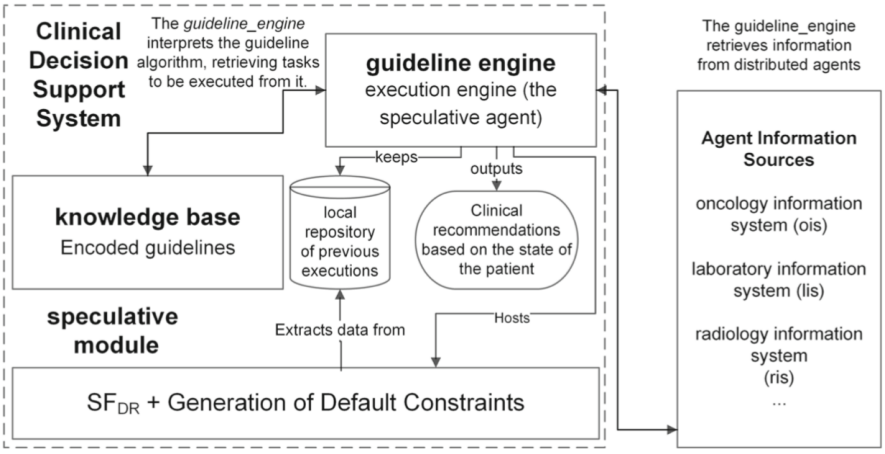

Fig. 6 Architecture of the clinical decision support system with its basic components, retrieved from Oliveira et al. (2017a)

To facilitate the creation, edition or removal of a clinical guideline, the system was enhanced with an editor based on the Protégé OWL API, which also provided a graphical user interface (Gonçalves et al. 2017).

Clinical Practice Guidelines provide recommendations for medical treatment including exams that should be done or have temporal constraints. These were introduced by Oliveira et al. (2017b), in which the temporal constraints were automatically interpreted and added to the physician's and patient's calendar. Thus, both of them could know in advance the next appointments or exams that should be executed. The system was tested against a dataset of colon cancer in order to prove its efficiency.

More recently, Oliveira et al. (2020) further improved CompGuide to include computational argumentation. This last step is useful when reasoning in cases of multimorbidity. When facing this scenario, the clinical decision process is more difficult, since the number of variables increases, optimization goals may be conflicting, and constraints may be impossible to be fully satisfied. Indeed, using an argumentation system, it is possible to achieve a solution that is consistent with the patient's multiple conditions and centered on the best goals for that patient.

5 Conclusions

Over the last years, the processing capacity of machines has evolved. Through them, it was possible to create more robust systems which included more functionalities. Indeed, nowadays the decision maker is supported by the results provided by decision support systems, which are able to consider a wider number of variables and, through machine learning algorithms, create connections between data that weren't possible to generate by humans.

There are different Artificial Intelligent algorithms, some of which may be more suitable in specific situations than others. These algorithms, with the huge volumes of data that are being generated, are able to achieve good results in a wide variety of fields, namely computer vision, optical character recognition, fraud detection, recommendation systems, among others. Despite the quality of such models, they lack in the ability to speculate a future yet unknown value, and thus prevent the execution of the model until such value is known. Speculative Computation is able to surpass this drawback, but it needs to have a default value set. For this process, different ML techniques may be used in order to improve the quality of the values that are considered as default, *i.e.*, generate a set that contains the most appropriate values to be considered when real information is missing.

The application of different techniques may enhance the features of Speculative Computation, which could be used in different scenarios. Previous sections described its application in two scenarios where the goals of each system considered distinct areas. The former introduced an adaptive localization system that, through virtual reality, guided the user during her/his outdoors walks. The path was adjusted to the user's specificities, and the speculative framework was able to anticipate user mistakes and alert in advance to try to prevent them from happening. The latter was used under a colon cancer scenario by interpreting clinical practice guidelines and adjust the treatment regarding the medical recommendations and patient health.

There are several Machine Learning algorithms and throughout this paper the goal was to demonstrate that the use of multiple techniques may enhance the final system and better adapt the results to the scenario it is being developed for. Indeed both cases presented show that, by applying trajectory data mining or Bayesian networks, the construction default revision set was improved and the system could benefit from better results provided by the advantages of the Speculative Computation framework.

Acknowledgments

This work has been supported by national funds through FCT - Fundação para a Ciência e a Tecnologia within the Project Scope: UIDB/00319/2020 and UIDB/04728/2020.

References

Adekitan, A. I., Abolade, J., and Shobayo, O. (2019). Data mining approach for predicting the daily Internet data traffic of a smart university. *Journal of Big Data*, 6(1):11.

Adeyemi, O. J., Popoola, S. I., Atayero, A. A., Afolayan, D. G., Ariyo, M., and Adetiba, E. (2018). Exploration of daily Internet data traffic generated in a smart university campus. *Data in Brief*, 20:30–52.

American Psychiatric Association (2013). *Diagnostic and Statistical Manual of Mental Disorders (DSM-5)*. 5 edition.

Boudol, G. and Petri, G. (2010). A Theory of Speculative Computation. In Gordon, A. D., editor, *Programming Languages and Systems*, pages 165–184, Berlin, Heidelberg. Springer Berlin Heidelberg.

Burton, F. W. (1985). Speculative computation, parallelism, and functional programming. *IEEE Transactions on Computers*, C-34(12):1190–1193.

Field, M. J. and Lohr, K. N., editors (1992). *Guidelines for Clinical Practice*. National Academies Press.

Fukuta, N., Satoh, K., and Yamaguchi, T. (2008). Towards "kiga-kiku" services on speculative computation. In Yamaguchi, T., editor, *Proceedings of the 7th International Conference on Practical Aspects of Knowledge Management (PAKM 2008), LNAI*, volume 5345, pages 256–267, Berlin, Heidelberg. Springer Berlin Heidelberg.

George, D. and Mallery, P. (2018). Logistic Regression. In *IBM SPSS Statistics 25 Step by Step*. Taylor & Francis.

Gewers, F. L., Ferreira, G. R., Arruda, H. F. D., Silva, F. N., Comin, C. H., Amancio, D. R., and Costa, L. D. F. (2021). Principal component analysis: A natural approach to data exploration. *ACM Comput. Surv.*, 54.

Gonçalves, F., Oliveira, T., Neves, J., and Novais, P. (2017). Compguide: Acquisition and editing of computer-interpretable guidelines. In Rocha, Á., Correia, A. M., Adeli, H., Reis, L. P., and Costanzo, S., editors, *Recent Advances in Information Systems and Technologies*, pages 257–266, Cham. Springer International Publishing.

Hosobe, H., Satoh, K., and Codognet, P. (2007). Agent-based speculative constraint processing. *IEICE TRANSACTIONS on Information and Systems*, E90-D:1354–1362.

Kakas, A. C. and Mancarella, P. (1990). On the relation between truth maintenance and abduction. pages 438–443.

Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516.

Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. (2020). Spectre Attacks: Exploiting Speculative Execution. *Commun. ACM*, 63(7):93–101.

Lanyi, C. S. and Brown, D. J. (2010). Design of Serious Games for Students with Intellectual Disability. In Joshi, A. and Dearden, A., editors, *IHCI10 Proceedings of the 2010 international conference on Interaction Design International Development*, pages 44–54. British Computer Society Swinton, UK.

Oliveira, T., Dauphin, J., Satoh, K., Tsumoto, S., and Novais, P. (2020). Goal-driven structured argumentation for patient management in a multimorbidity setting. In Dastani, M. and van der

Torre, H. D. L., editors, *Logic and Argumentation, 3rd Internatinoal Conference, CLAR 2020, LNAI21061*, pages 166–183, Cham. Springer International Publishing.

Oliveira, T., Neves, J., Novais, P., and Satoh, K. (2014). Applying speculative computation to guideline-based decision support systems. In *Proceedings of the 2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, CBMS '14, pages 42–47, USA. IEEE Computer Society.

Oliveira, T., Novais, P., and Neves, J. (2013). *Representation of Clinical Practice Guideline Components in OWL*, volume 221. Springer Verlag.

Oliveira, T., Satoh, K., Novais, P., Neves, J., and Hosobe, H. (2017a). A dynamic default revision mechanism for speculative computation. *Autonomous Agents and Multi-Agent Systems*, 31:656–695.

Oliveira, T., Silva, A., Neves, J., and Novais, P. (2017b). Decision support provided by a temporally oriented health care assistant: An implementation of computer-interpretable guidelines. *Journal of Medical Systems*, 41.

Pisner, D. A. and Schnyer, D. M. (2020). Chapter 6 - Support vector machine. In Mechelli, A. and Vieira, S., editors, *Machine Learning*, pages 101–121. Academic Press.

Pun, L., Zhao, P., and Liu, X. (2019). A Multiple Regression Approach for Traffic Flow Estimation. *IEEE Access*, 7:35998–36009.

Ramos, J., César, A., Neves, J., and Novais, P. (2017a). Adapting the user path through trajectory data mining. In Paz, J. F. D., Julián, V., Villarrubia, G., Marreiros, G., and Novais, P., editors, *Ambient Intelligence– Software and Applications – 8th International Symposium on Ambient Intelligence (ISAmI 2017)*, pages 195–202, Cham. Springer International Publishing.

Ramos, J., Oliveira, T., Satoh, K., Neves, J., and Novais, P. (2017b). An orientation method with prediction and anticipation features. *Inteligencia Artificial*, 20:82.

Ramos, J., Oliveira, T., Satoh, K., Neves, J., and Novais, P. (2018). Cognitive assistants-an analysis and future trends based on speculative default reasoning. *Applied Sciences (Switzerland)*, 8.

Ronao, C. A. and Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235–244.

Satoh, K. (2005). Speculative computation and abduction for an autonomous agent. *IEICE - Trans. Inf. Syst.*, E88-D:2031–2038.

Satoh, K., Inoue, K., Iwanuma, K., and Sakama, C. (2000). Speculative computation by abduction under incomplete communication environments. In *Proceedings Fourth International Conference on MultiAgent Systems*, pages 263–270.

Satoh, K. and Yamamoto, K. (2002). Speculative computation with multi-agent belief revision. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, AAMAS '02, pages 897–904, New York, NY, USA. Association for Computing Machinery.

Schalock, R. L., Borthwick-Duffy, S. A., Bradley, V. J., Buntinx, W. H. E., Coulter, D. L., Craig, E. M., Gomez, S. C., Lachapelle, Y., Luckasson, R., Reeve, A., Shogren, K. A., Snell, M. E., Spreat, S., Tasse, M. J., Thompson, J. R., Verdugo-Alonso, M. A., Wehmeyer, M. L., and Yeager, M. H. (2010). *Intellectual Disability: Definition, Classification, and Systems of Supports. Eleventh Edition*. American Association on Intellectual and Developmental Disabilities.

Schonlau, M. and Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, 20(1):3–29.

Vadla, P. K., Ruwali, A., Prakash, K. B., Lakshmi, M. V. P., and Kanagachidambaresan, G. R. (2021). *Neural Network*, pages 39–43. Springer International Publishing, Cham.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018:7068349.

Xie, H., Zhang, L., Lim, C. P., Yu, Y., Liu, C., Liu, H., and Walters, J. (2019). Improving k-means clustering with enhanced firefly algorithms. *Applied Soft Computing*, 84:105763.