Daniela Paliaga – Vincenzo de Lisi

L'algoritmo nel contenzioso giuslavoristico

Premessa

La capillare presenza dell'informatica nella società industrializzata in cui viviamo è evidente.

Un incredibile numero di operazioni, che un tempo era svolto dall'uomo, è oggi affidato alle "macchine", appositamente istruite dall'uomo attraverso algoritmi più o meno sofisticati.

E alle "macchine" l'uomo sta insegnando anche a pensare, ad apprendere da sole nel corso del loro lavoro.

L'esempio più immediatamente percepibile di tutto ciò si rinviene nel commercio elettronico: una qualsiasi ricerca su internet si traduce immancabilmente in successive proposte di acquisto, dietro le quali non ci sono persone che analizzano ciò che facciamo, ne estraggono informazioni sui nostri interessi, cercano prodotti che possano soddisfarli, li selezionano e ce li propongono, bensì un sofisticato algoritmo.

Moltissimi passaggi della vita di ogni giorno vedono operare analoghi algoritmi: gli assistenti virtuali come siri, ehi google e alexa; quarantine che filtra le email sospette; netflix che ci consiglia nuovi film analizzando le nostre scelte precedenti; il correttore automatico di word o whatsapp sono soltanto alcuni degli esempi a noi più vicini.

Da tempo sono allo studio esempi ben più eclatanti che sembrano ancora ai confini della fantascienza, ma non lo sono affatto, come i sistemi di guida automatica o la giustizia predittiva.

Tutto ciò è inevitabilmente destinato ad arrivare "sul tavolo" del giudice e richiede uno sforzo consistente per acquisire gli strumenti minimi che consentano di comprendere questa nuova realtà e orientarsi al suo interno nell'attività di accertamento e valutazione richiesta dalla decisione della controversia.

L'obiettivo della presente relazione "integrata" è quello di "accendere i riflettori" sulle ricadute che tutto ciò ha nel lavoro del giudice e, in particolare, sui problemi processuali relativi all'accertamento di come funziona l'algoritmo, suscitando quella consapevolezza delle difficoltà in esso insite che è la premessa indispensabile di un corretto approccio professionale a questa nuova realtà. A tale scopo, al par. 6, Vincenzo De Lisi tratteggia un quadro d'insieme del mondo a cui – per esigenze di sintesi, ma spesso anche per ignoranza – facciamo ormai riferimento parlando di algoritmo, illustrando alcuni concetti informatici fondamentali e fornendo preziose informazioni su come nascono e funzionano gli algoritmi tratte dalla sua esperienza professionale di ingegnere informatico.

Partendo da tali spunti essenziali, nel resto della relazione Daniela Paliaga sviluppa alcune riflessioni destinate a costituire una prima traccia per il giudice del lavoro che si trovi a valutare: l'adeguatezza delle allegazioni e delle prove offerte dalle parti in ordine al funzionamento dell'algoritmo, se e come esercitare i poteri d'ufficio e, infine, l'efficacia probatoria di ciò che è presente in atti al termine dell'istruttoria.

1. L'algoritmo ed il rapporto di lavoro

La sostituzione dell'uomo con sistemi informatici che eseguono algoritmi è ormai una realtà anche nel mondo del lavoro: le aziende se ne servono per produrre¸ ma anche per gestire tutta una serie di passaggi del rapporto di lavoro. I sistemi informatici sono in grado di raccogliere e archiviare una quantità impressionante di dati anche minimi relativi all'attività lavorativa svolta dal dipendente. Si tratta, ad esempio, del tempo impiegato per svolgere le varie operazioni che compongono la sequenza produttiva con dispositivi elettronici, delle espressioni di gradimento della clientela, di numero e natura degli accessi ad internet di un impiegato, del percorso seguito per raggiungere una certa destinazione dal dipendente munito di geolocalizzatore…..

Con altrettanta facilità essi possono organizzare ed elaborare i dati a loro disposizione – siano stati raccolti da essi stessi oppure messi a loro disposizione dall'esterno – per trarne conclusioni richieste dal datore di lavoro.

L'uso che il datore di lavoro può fare di questi dati e della loro elaborazione è il più disparato: possono servire a premiare e a sanzionare, a scegliere coloro che vanno promossi, a trasferire, a licenziare.

È sufficiente incaricare ad uno o più programmatori di creare appositi algoritmi che affidino al sistema informatico una certa operazione esprimendo con apposito linguaggio di programmazione le istruzioni che esso deve seguire per raggiungere il risultato richiesto.

In tal modo il datore di lavoro chiede alla macchina di fare ciò che un tempo facevano egli stesso o i suoi dipendenti, girando per la fabbrica, osservando i lavoratori e analizzando in modo "umano" i risultati della produzione. L'uso della macchina consente di moltiplicare i passaggi di questo processo di rilevazione, raccolta e elaborazione del dato ed accelerarlo in misura impressionante rispetto a ciò che può fare l'uomo, fosse anche una squadra di persone a ciò dedicata.

Ciò che può fare un software oggi realizza il sogno nascosto di ogni datore di lavoro di esercitare un controllo totale e capillare del dipendente, compiere un'analisi approfondita della sua attività lavorativa e di ogni altro aspetto ritenuto rilevante (anche raccolto su internet e relativo alla vita personale) e prendere le decisioni conseguenti in modo "matematico" e "oggettivo".

2. Profili problematici, non solo per il lavoro

Se questo sia davvero e sempre un aiuto per l'imprenditore non è scontato.

Il primo rischio generale è che, nella fase decisionale, il datore di lavoro rinunci del tutto alla valutazione «umana» – quella che si giova dell'intuizione e dell'esperienza di chi, per professione, gestisce il personale – e perda così l'opportunità di valorizzare tutta una serie di aspetti che la macchina e i controlli attuati attraverso di essa, seppure certosini, potrebbero non rilevare e che, però, possono essere determinanti per una valutazione completa e utile del lavoratore. Caratteristiche come, ad esempio, la buona relazione con i colleghi, la disponibilità o la capacità di adattarsi possono essere valorizzate in base ai risultati oggettivi soltanto in misura molto modesta, mentre possono essere ben apprezzati da chi abbia voglia di osservare con i propri occhi il comportamento umano.

C'è un secondo grave rischio.

L'idea più diffusa tra i non tecnici è probabilmente che l'algoritmo raggiunge il suo obiettivo molto meglio del cervello umano, non solo perché è incredibilmente più veloce e non ha alcun problema di capienza della sua concentrazione e memoria, ma anche perché è obiettivo e imparziale.

Da tempo, però, i tecnici si sono accorti che anche gli algoritmi sbagliano. Può trattarsi innanzi tutto delle cd. anomalie software, che si hanno quando, per errore tecnico, il codice in produzione (cioè nell'ambiente di esecuzione finale) si comporta in modo diverso da quanto atteso. In tal caso l'impresa, una volta identificato il comportamento errato – cosa che spesso avviene grazie a segna-

lazioni di utenti esterni o interni – mette in atto delle attività di adeguamento del codice per sistemare il problema (cd. manutenzione correttiva). Ma gli algoritmi possono anche essere vittime dei cd. bias (termine inglese per pregiudizi algoritmici), cioè errori sistematici di giudizio o di interpretazione che possono generarsi nel processo di autoapprendimento.

In questo caso non si tratta di un errore software, da malfunzionamento del codice (che si ha ad esempio se un algoritmo disegnato per fare una somma, per errore di scrittura del codice, effettua una sottrazione), ma di errore nelle sue regole.

Nel mondo dell'intelligenza artificiale, in particolare, può trattarsi di errore nel suo addestramento.

Le applicazioni della cd. intelligenza artficiale sono ormai moltissime: andiamo da quelle relativamente semplici della nostra vita quotidiana (ad esempio, il software di Netflix che sceglie cosa suggerirci in base alle scelte precedenti) a quelli molto più complessi (come, ad esempio, i software usati nell'ambito dei programmi spaziali).

Ebbene, in questo contesto è possibile che, nel suo processo di autoapprendimento, l'algoritmo commetta degli errori e cioè generi istruzioni errate, che tradiscono lo scopo affidatogli.

Ciò avviene sulla base dei dati che esso elabora (cd. dataset).

Gli algoritmi di intelligenza artificiale, infatti, sono sistemi di autoapprendimento, con modelli matematici, che si basano sul dato. Le caratteristiche dei dati che vengono forniti possono influenzarlo. Se l'insieme dei dati forniti contiene in sé un pregiudizio razziale, di genere, o di altro tipo, è molto probabile che l'algoritmo si comporti in modo pregiudizievole. In questo caso non è un problema del modello, ma del fatto che ad esso, durante la fase di addestramento o nella sua "vita" successiva, sono stati forniti dati distorti o non completi.

È accaduto, ad esempio, nel campo del riconoscimento facciale – quello usato per sbloccare lo smartphone o sostituire i controlli umani alle frontiere o negli aeroporti – in cui, nel 2018, si è scoperto che tre programmi di importanti aziende tecnologiche incorporavano pregiudizi razziali e di genere e, in particolare, arrivavano ad un errore di riconoscimento delle donne con pelle scura tra il 20 e il 34 %, mentre per gli uomini di pelle bianca l'errore non superava lo 0,8%.

Un esempio emblematico nel mondo del lavoro riguarda il software di recruitment utilizzato da Amazon a partire dal 2014 per analizzare i curricula dei candidati ed automatizzare la procedura di selezione. Il sistema penalizzava le donne in quanto era stato addestrato con dati reali contenenti i CV ricevuti dalla società nei 10 anni precedenti e, trattandosi di ruoli tecnici prevalentemente svolti da uomini, aveva individuato come caratteristica ideale del dipendente Amazon il genere maschile.

La consapevolezza dell'insidia costituita dai bias ha dato sviluppo ad un settore di studi interdisciplinari denominato di "Algorithmic fairness" che ha lo scopo di mitigare gli effetti dei pregiudizi e delle discriminazioni ingiustificate sugli individui che possono insorgere nell'apprendimento automatico attraverso la creazione di modelli di apprendimento in grado di effettuare previsioni corrette. Negli USA a dicembre 2020 un senatore democratico ha proposto il "Algorithmic Fairness Act" volto ad ottenere dalla Federal Trade Commission regole che garantiscano che gli algoritmi vengano utilizzati in modo equo e che coloro che sono interessati dalle loro decisioni siano informati e abbiano l'opportunità di correggerne gli errori.

Il Consiglio d'Europa l'8 aprile 2020 ha emesso una raccomandazione agli stati membri in ordine agli "impatti sui diritti umani dei sistemi algoritmici".

Il 19 febbraio 2020 la Commissione europea ha pubblicato il "Libro bianco sull'intelligenza artificiale", in cui definisce le opzioni strategiche destinate al duplice obiettivo di promuovere l'adozione dell'IA e affrontare i rischi associati a determinati utilizzi di tale tecnologia e il 21 aprile 2021, in attuazione di questo secondo obiettivo, ha pubblicato una "Proposta di regolamento del parlamento europeo e del consiglio che stabilisce regole armonizzate sull'intelligenza artificiale" destinate a sviluppare un ecosistema di fiducia proponendo un quadro giuridico per un'IA affidabile sull'approccio europeo all'intelligenza artificiale.

Non è detto che il datore sia sempre interessato ad evitare tali errori. Amazon, ad esempio, nel 2015 dismise il sistema informatico di reclutamento del personale sopra menzionato che aveva elaborato il pregiudizio di genere, ma online si può anche leggere, con riferimento al servizio Flex affidato a driver indipendenti, che "Stando agli ex manager, Amazon sapeva che delegare il lavoro alle macchine avrebbe portato a errori, ma ha deciso che era più economico fidarsi degli algoritmi piuttosto che pagare le persone per indagare su consegne errate. Del resto, i driver possono essere sostituiti facilmente".

Di sicuro, però, tutto ciò può integrare un grave pregiudizio per i lavoratori.

L'uso dell'algoritmo per prendere delle decisioni, peraltro, potrebbe anche nascondere comportamenti volutamente scorretti del datore di lavoro il quale, dietro il paravento di un'analisi oggettiva della produzione e di scelte volte ad ottimizzarla, nasconda in realtà analisi e controlli relativi ad altri aspetti e scelte di natura discriminatoria.

Un datore di lavoro che debba porre in essere un licenziamento collettivo, ad esempio, potrebbe essere tentato di infilare nell'algoritmo che sceglie i lavoratori da sacrificare, tra i criteri di legge o concordati anche qualche criterio utile a «bonificare» almeno in parte l'organico aziendale da soggetti meno graditi, ad esempio, per appartenenza al sindacato.

3. Le prime esperienze giurisprudenziali

L'algoritmo è già arrivato sul tavolo del giudice del lavoro in due principali tipologie di controversie.

La più risalente riguarda la procedura di mobilità straordinaria dei docenti dell'anno 2016-2017 attraverso la quale fu data una collocazione definitiva a tutti coloro che erano stati assunti con la procedura straordinaria di assunzione cd. della buona scuola.

I docenti contestavano i risultati della mobilità evidenziando come, nell'attribuzione delle sedi, erano stati preferiti docenti che avevano un punteggio inferiore.

L'illogicità della situazione fece parlare di algoritmo impazzito, ma non era affatto così e, a quanto pare, non si trattava neanche di un bias. Semplicemente l'algoritmo era stato impostato male, indicando come criterio prioritario, vincente anche sul punteggio, il fatto che una certa sede fosse la prima scelta dal candidato.

Più recente, e tutta da esplorare, è la presenza dell'algoritmo nelle controversie con cui i rider affermano la natura subordinata del rapporto che li lega alla piattaforma allegando tutta una serie di comportamenti dell'algoritmo che integrerebbero un puntuale esercizio del potere direttivo, di controllo e sanzionatorio tipici della subordinazione.

4. Il contenzioso che ci attende

La vita reale offre la notizia di svariati altri impieghi dell'algoritmo nel mondo del lavoro che, prima o poi, arriveranno anche nelle aule di giustizia. Secondo alcuni studi statistici il numero di aziende che adottano tecniche di Intelligenza Artificiale è cresciuto del 270% negli ultimi 4 anni (studio Oberlo) e, nel 2019, un'azienda su tre sfruttava l'Intelligenza Artificiale o aveva intenzione di farlo (studio Gartner) .

Amazon è certamente un'impresa che utilizza massicciamente informatica ed algoritmi non soltanto nei rapporti con la clientela, ma anche in ogni passaggio dell'attività lavorativa e nella gestione del rapporto con dipendenti e altri collaboratori.

A marzo 2021 gli autisti di Amazon hanno scioperato per protestare contro i ritmi di lavoro impressi dall'algoritmo che non tengono in alcun conto le situazioni contingenti come, ad es., le condizioni di traffico o climatiche.

Da notizie di stampa provenienti dagli Stati Uniti apprendiamo che è l'algoritmo a scegliere chi viene licenziato da Amazon. Gli algoritmi monitorano ogni aspetto del lavoro dei conducenti – come, ad esempio, il rispetto dei tempi

e delle indicazioni di consegna – analizzano i dati per individuare modelli di prestazioni e poi decidono quali di essi lavorano meglio e quali vanno lasciati a casa. Praticamente non c'è feedback "umano" per i conducenti, i quali ricevono a fine giornata valutazioni sintetiche come fantastico, ottimo, giusto, a rischio e solo occasionalmente e-mail automatizzate.

La pubblica amministrazione, d'altronde, utilizza ormai regolarmente l'algoritmo per automatizzare i procedimenti, in particolare per gestire le procedure concorsuali e ciò è del tutto comprensibile visto l'enorme numero di posizioni coinvolte, soprattutto quando ciò avviene su base nazionale.

Se ne è occupata di recente la giustizia amministrativa in alcune recenti decisioni in materia di diritto di accesso.

Partendo dalla considerazione che la decisione amministrativa algoritmica è comunque una decisione amministrativa – rispetto alla quale deve essere garantita la possibilità del destinatario di verificarne la legittimità e del giudice amministrativo di esercitare il sindacato giurisdizionale con cognizione piena sulla stessa, anche sotto il profilo della sua logicità e ragionevolezza – il giudice amministrativo ha affermato il diritto alla conoscibilità del meccanismo attraverso cui si giunge alla decisione amministrativa, anche quando essa si avvale dell'automazione.

Superando le eccezioni del Ministero dell'Istruzione basate sul pregiudizio che essa avrebbe arrecato alla proprietà intellettuale del software da parte del CINECA (il Consorzio interuniversitario che, sotto il controllo dello stesso Ministero, fornisce beni e servizi informatici a numerosi enti pubblici ed è stato qualificato come controinteressato dalla sentenza n. 30/2020 del Consiglio di Stato) ed alla possibilità di riutilizzare il programma, il TAR Lazio[1] ha dunque disposto l'ostensione del codice sorgente del software utilizzato nel caso di specie per automatizzare alcune fasi di un concorso pubblico per dirigenti scolastici indetto dal Ministero dell'Istruzione.

Di certo l'uso dell'algoritmo da parte del datore di lavoro può di per sé essere fonte di violazione dei diritti del lavoratore sotto vari profili.

Ne abbiamo già i primi esempi in tema di tutela della privacy e di discriminazione.

Sotto il primo profilo è di grande interesse quanto emerso nel corso dell'indagine esperita dall'autorità garante della privacy sull'algoritmo utilizzato da alcune società che gestiscono piattaforme di consegna dei pasti a domicilio e che ha condotto alle ordinanze ingiunzione nei confronti di Foodinho s.r.l. del 10 giugno 2021 e di Deliveroo Italy s.r.l. del 22 luglio 2021.

[1] Tra le più recenti Tar Lazio nn. 7370/2020, 7526/2020 e 13692/2020.

Un primo esempio di problematica relativa alla discriminazione attuata attraverso l'algoritmo è costituito, invece, dall'ordinanza del Tribunale di Bologna del 31 dicembre 2020 (giudice Chiara Zompi) con cui l'algoritmo utilizzato per gestire le prenotazioni delle sessioni di lavoro da parte dei rider è stato ritenuto discriminatorio.

Difficile escludere, poi, che prima o poi dovremo occuparci della discriminazione realizzata dai bias o dei problemi di trasparenza e controllabilità da parte di lavoratore, sindacato e giudice che derivano dall'uso dell'algoritmo in contesti in cui (come, ad esempio, in caso di licenziamento collettivo o di trasferimento d'azienda) la tutela del lavoratore viene affidata alla procedimentalizzazione del potere datoriale.

La presente relazione si concentra su un aspetto comune a tutte le possibili controversie in cui la decisione dipende, in tutto o in parte, dall'accertamento del se e come ha operato un algoritmo, e cioè il problema di come procedere concretamente a tale accertamento.

5. Il problema dell'accertamento

Qualunque sia l'oggetto della controversia in cui compare l'algoritmo, il giudice del lavoro si trova di fronte ad un problema processuale di carattere generale: come si può accertare come ha funzionato e/o funziona un algoritmo? Tale accertamento richiede competenze tecniche elevate che il giudice di certo non ha.

Lo strumento a cui il giudice solitamente ricorre, nei casi in cui la sua decisione necessita di specifiche competenze tecniche estranee alla sua formazione, è la CTU.

A volte queste competenze non servono soltanto per affrontare il quantum della decisione (ad esempio, per valutare il danno alla salute derivato da un infortunio), ma investono l'intera decisione (ad esempio la natura professionale di una malattia) e vanno dunque spese anche per decidere sull'an della domanda. In questi casi è quantomai opportuno farsi assistere dal CTU sin dall'inizio, come consente l'art. 61 c.p.c., per chiedergli ausilio nelle scelte istruttorie e cioè – una volta risolti i problemi di ammissibilità delle istanze di parte e delle iniziative d'ufficio, che ovviamente sono e restano di esclusiva competenza del giudice – per distinguere le istanze istruttorie utili da quelle inutili e, eventualmente, suggerire quali iniziative d'ufficio possano servire.

Come emerge chiaramente dal par. 6, quando c'è di mezzo l'algoritmo, le cose stanno ben diversamente, in quanto ricorrere al CTU è tutt'altro che facile e risolutivo.

La lacuna delle nostre competenze va dunque affrontata prima e fuori del processo e, vista la diffusione dell'informatica nel mondo del lavoro, ciò va fatto anche con una certa urgenza.

Essendo illusorio pensare di poter recuperare queste conoscenze da soli, tanto più nello spazio limitato dello studio del singolo fascicolo, appare in tutta la sua evidenza l'importanza della formazione e la necessità che essa operi su due fronti: da un lato, offrire un minimo di conoscenze di base di natura informatica e, dall'altro, iniziare a riflettere sull'applicazione delle regole processuali a questo mondo particolare.

Sotto il primo profilo risulta di grande utilità sottoporre all'ing. De Lisi alcuni quesiti di fondo affinché, come si accennava all'inizio, le sue risposte ricostruiscano le linee essenziali di ciò a cui facciamo riferimento parlando di algoritmo e ci consentano di comprendere quali sono i passaggi e i documenti a cui il giudice può fare riferimento per accertare come ciò che continuiamo per comodità a chiamare "l'algoritmo" ha effettivamente funzionato in concreto, ogniqualvolta che ciò sia oggetto di controversia tra le parti.

L'interrogativo di fondo è "di cosa stiamo parlando?" e, in base alle frammentarie informazioni già in nostro possesso, può essere declinato in varie domande più puntuali: cos'è un algoritmo? cos'è un programma? come sono legati tra loro? cos'è e come funziona l'intelligenza artificiale?

6. Aspetti tecnici utili (Vincenzo De Lisi)

Un algoritmo è un insieme di istruzioni in linguaggio comune che devono essere eseguite per risolvere un problema o raggiungere un risultato. Ognuno di noi esegue algoritmi, ad esempio quando cucina.

Perché l'algoritmo funzioni, le istruzioni devono essere eseguite esattamente nell'ordine in cui sono espresse. Alcune di esse sono condizionali e cioè variano a seconda che si realizzi o meno una certa condizione.

L'algoritmo per preparare il tè, ad esempio, contiene le seguenti istruzioni: 1) scalda l'acqua, 2) sistema un filtro in una tazza, 3) versa nella tazza l'acqua calda, 4) lascia in infusione per qualche minuto, 5) aggiungi zucchero, 6) controlla se il te è abbastanza dolce (regola condizionale): se no ripeti 5), se sì, il tè è pronto.

Un programma non è altro che la traduzione di un algoritmo tramite un linguaggio di programmazione in un cd. codice sorgente, che viene eseguito dal computer. Per poter essere compreso ed eseguito dal computer il codice, composto da istruzioni testuali scritte in un determinato linguaggio di programmazione, può essere direttamente interpretato (generalmente dentro un

browser) oppure richiede un passaggio per renderlo eseguibile, come nel caso degli applicativi che girano sul nostro computer (word, consolle, chrome, …). In letteratura esistono diverse centinaia di linguaggi di programmazione, nuovi e vecchi, molti usati per specifici casi d'uso. Un programma commerciale è generalmente composto da diversi componenti (realizzano un'architettura, come in un palazzo). Ogni componente può essere scritta in diversi linguaggi. Per lo sviluppo della soluzione si usano anche parti di codice esterne che si trovano (open source o a pagamento) sotto forma di librerie o framework comuni per semplificare il lavoro di uno sviluppatore, ma si tratta sempre di codice.

Quando si voglia accertare in sede giudiziale "come funziona un algoritmo", è essenziale avere chiaro che il tema da cui partire non è il codice. Le sue dimensioni e la sua complessità, infatti, renderebbero impossibile il lavoro di analisi.

La via corretta da seguire risulta evidente ove si ricordi che un software è la traduzione di un algoritmo e l'algoritmo è la soluzione ad un problema. Per realizzare un software, dunque, si parte sempre da una fase di raccolta dei cd. "requisiti utente", che spiegano cosa deve fare il software "lato utente", e si passa poi alla fase di progettazione in cui si descrive come il software deve implementare il requisito utente.

Senza queste informazioni uno sviluppatore non è in grado di realizzare il software. Lo sviluppatore, infatti, deve sapere cosa il software deve fare. Ciò comporta che, da qualche parte, esiste per forza un supporto documentale in cui, in modo più o meno formale, si trovano tali informazioni cioè i requisiti. Nelle aziende di startup o di piccole dimensioni, formate da sviluppatori che realizzano soluzioni, nella fase iniziale il software potrebbe non avere un adeguato supporto documentale dei requisiti, ma anche in tal caso quest'ultimo diventa indispensabile nelle fasi successive.

Non ci sono regole definite per memorizzare questi requisiti, si può andare da Word o Excel fino a strumenti specifici per la gestione del ciclo di vita del software che permettono di tenere traccia di molte attività, tra le quali i requisiti. Per questo aspetto li possiamo vedere come un database dove memorizziamo i requisiti (che non sono altro che testo).

Come si è accennato, esso presenta varie componenti, alcune delle quali vengono appositamente scritte mentre altre possono essere acquisite già pronte, in quanto scritte da altri, e possono anche essere open source.

Non sono certo tali quelle che si rinvengono negli algoritmi delle grandi aziende come Uber, in quanto tali algoritmi sono il cuore della loro attività e dunque tendono ad essere "proprietari" (detti anche non liberi o closed source), cioè protetti con il copyright.

Accertare come funziona un determinato software non è un'operazione facile, neanche per un tecnico, e ciò per varie ragioni.

Un primo fattore di complicazione nell'accertamento è dato dal fatto che esiste una molteplicità di linguaggi di programmazione (centinaia) e che questi evolvono con grande frenesia, guidati dalle evoluzioni tecnologiche.

È un mondo che cambia ad una velocità enorme.

Molti linguaggi hanno indubbiamente delle similitudini di logica, in quanto appartengono a tipologie di linguaggi simili (linguaggi ad oggetti, relazionali, funzionali, procedurali, …), ma se si deve affrontare un codice complesso, è importante avere l'esperto di quel determinato linguaggio.

A differenza del mondo dell'impresa tradizionale o della PA – dove il cambiamento tecnologico ha logiche di business e di mercato strutturalmente più lente e troviamo spesso linguaggi con una forte "tradizione" – le moderne start up, peraltro, utilizzano spesso linguaggi di ultima generazione.

Un bravo sviluppatore conosce sicuramente molti linguaggi in base alla sua esperienza o al suo ruolo, ma a causa della vastità delle tecnologie essi sono solo una parte di tutti quelli utilizzabili e concretamente in uso in un determinato momento storico.

Ad esempio, uno sviluppatore bravo nelle interfacce utente (cioè in quello con cui interagiamo in una maschera di un applicativo o una pagina web), in media utilizzerà dei linguaggi diversi da chi è specializzato nelle logiche di business (si intende la logica principale, l'algoritmo) e/o nella gestione dei dati (database, dove posso mettere del codice per la manipolazione di un dato).

Il giudice che debba cercare un CTU per affidargli la lettura del codice sorgente o la ricostruzione dell'algoritmo che sta dietro un programma deve essere ben consapevole della difficoltà di individuare la persona giusta data dal fatto che non esiste un esperto informatico, per quanto bravo, che sappia tutto. Ed in particolare deve essere consapevole che potrebbe essere una strada che non porta a nulla, in quanto le dimensioni e le relazioni tra parti di codice, senza adeguata documentazione a supporto, lo rendono non leggibile.

Ciò è di tutta evidenza quando si pensa ai codici di milioni di righe, solitamente scritti da molti sviluppatori con molti linguaggi di programmazione diversi, ma non è soltanto una questione di numero di righe di codice. Tra i tanti fattori che possono rendere un codice più o meno facile o complesso c'è, ad esempio, la qualità dello sviluppatore, il fatto che questi lo scriva bene, usando "nomi leggibili" e una logica "pulita".

Il reverse engineering[2] di un codice complesso potrebbe facilmente richiedere una squadra di tecnici esperti ed un tempo indefinito, oltre ai costi per l'acquisto di strumenti a supporto.

[2] Che consiste nel processo di analisi di un sistema software esistente, eseguito al fine di crearne una rappresentazione ad alto livello di astrazione.

Un altro motivo di grande complicazione è dato dal fatto che il software è soggetto a continui aggiornamenti.

Il codice viene aggiornato costantemente, per novità, modifiche, correzioni, aggiunte, normative. Ci sono team di sviluppatori sempre attivi. Anche nel nostro caso (Ministero della Giustizia) le aziende che lavorano per noi producono costantemente tanto codice al giorno.

Le aziende strutturate utilizzano dei sistemi di "versioning" del codice che permettono di risalire ad una determinata versione. Anche in questo caso, tuttavia, la questione è complessa perché, come già detto, un software può essere composto da diverse parti, ognuna con un suo percorso di aggiornamento, e di conseguenza di versionamento. Un codice sviluppato da diverse persone insieme segue sicuramente questi meccanismi, ma il loro recupero nell'ambito di un accertamento dipende, come per il resto, dal modello di lavoro adottato dell'azienda.

Ad oggi, in contesti evoluti, è prassi un modello di rilascio del software (cioè l'operazione che si realizza quando una modifica viene portata in produzione) on-demand ed automatico. Questo vuol dire che è possibile trovare diverse modifiche al codice o a sue parti nell'arco di una stessa giornata. Il punto chiave, anche in questo caso, è risalire alla documentazione che spiega il perché di una modifica, cercando di determinare (se necessario) la logica del codice.

Un altro aspetto cruciale nell'ambito del cd. reverse engineering è costituito dall'importanza del dato nel funzionamento dell'algoritmo e deriva dal fatto che, come si è accennato, nel settore che qui interessa l'algoritmo viene solitamente utilizzato per elaborare dati.

Ciò comporta che esso, e di conseguenza il programma realizzato per seguirne le regole, si basa sempre sui dati (input) a disposizione in un determinato momento e può dunque cambiare il suo risultato (output) in ogni momento in base ai dati che riceve.

L'algoritmo di Google Maps, ad esempio, decide il percorso migliore, non solo sulla base della distanza, ma anche in base al traffico presente in quel dato momento.

Si tratta di un aspetto che è essenziale mettere a fuoco quando se ne voglia accertare il funzionamento: ogni algoritmo si comporta in un certo modo in base ai dati che riceve.

Nel mondo dell'intelligenza artificiale (cd. AI), il dato è ancora più importante. Come è stato accennato nella parte introduttiva, nel mondo dell'AI vengono utilizzate molteplici tipologie di algoritmi, semplici e complessi, ma il tema centrale è quello dei dati utilizzati anche ed innanzi tutto per addestrarli. Spesso, in effetti, nelle fasi realizzative di un progetto di AI, la fase di training è prevalente rispetto al tempo di sviluppo dell'algoritmo.

Per addestrare gli algoritmi di intelligenza artificiale, la qualità del dato e la sua completezza anche lato statistico, sono l'elemento vitale, per gestire il rischio del Bias.

Per le ragioni sinora esposte è evidente che, per comprendere il funzionamento dell'algoritmo, non basta avere il suo codice sorgente.

Senza informazioni aggiuntive, il codice è molto difficile da leggere. Se è complesso, è quasi impossibile.

Per capire cosa fa un codice complesso anche uno sviluppatore bravo ha necessità di strumenti e informazioni aggiuntive. Senza un supporto documentale adeguato, che spiega le logiche del codice, il suo comportamento atteso e la sua strutturazione, infatti, l'analisi risulta molto lunga e complessa. Ricordiamo, come già descritto, che un programma software è un insieme di un gran numero di parti di codice che interagiscono tra di loro, che possono essere librerie di classi interne o esterne (open source o commerciali), framework di sviluppo, componenti distribuite su architetture distribuite su diversi sistemi, strati di codice che dialogano con altri software, dove ogni pezzo di codice può essere scritto in uno o più linguaggi di programmazioni, in base alla necessità degli sviluppatori.

Questo rende essenziale focalizzare il lavoro di accertamento dapprima nell'individuare quella determinata parte di algoritmo di interesse, poi nel ricercare la documentazione ad esso associata ed eventualmente anche una porzione di codice e, quindi, nel ricreare il contesto di operatività del software.

Avere la porzione di codice ed i relativi requisiti non basta, perché per far funzionare in autonomia quella singola parte del codice si deve anche ricreare il suo ambiente (software, hardware, database, – …. relazioni con il resto del codice) ed anche i dati di contesto.

Nel mondo di AI la situazione cambia perché, come descritto nella prima parte, il tema è l'addestramento dei dati ed il modello matematico utilizzato. Il focus primario è sui dati usati per l'addestramento.

Ovviamente, quando si tratta di programmi complessi e l'accertamento riguarda soltanto uno o più passaggi ben definiti, è essenziale limitare l'operazione soltanto a questi ultimi. Ma anche in tal caso l'operazione rimane complessa in quanto il codice sorgente "invoca" altri pezzi di codice e/o ha necessità di dati in input o altre informazioni (ad esempio, per i rider, la posizione del cellulare, i tempi del ristorante, la posizione del destinatario…).

A questo riguardo è essenziale la collaborazione di chi ha creato il software. Costoro, infatti, sono i primi ad avere la necessità di provare il funzionamento delle singole parti di codice e dunque hanno sicuramente un ambiente di prova in cui si possono far "girare" i singoli pezzi di esso, costruendo intorno ad essi tutto ciò che serve per la simulazione.

Per tutte le ragioni esposte, più che sul codice sorgente, è importante focalizzarsi sulle logiche di progettazione ed i requisiti. È in questo ambito che un CTU esperto di progettazione può avere maggiore successo, in quanto le metodologie di progettazione e di definizione dei requisiti, se pur molteplici, hanno dei principi comuni.

Il codice, al limite, può essere usato per integrare la prova fornita dai requisiti, ove essa risulti insufficiente e l'algoritmo non sia troppo complesso. La lettura dei requisiti da parte di un esperto, peraltro, consentirebbe di affinare la ricerca all'interno del codice ed aumentare le sue possibilità di successo.

È quasi impossibile, invece, ricostruire la modalità di funzionamento del software dal solo codice, senza documenti di appoggio.

7. Torniamo al profilo giuridico

I concetti e le informazioni di fondo ora illustrati costituiscono la premessa essenziale per svolgere alcune considerazioni sul tema processuale dell'accertamento di come funziona l'algoritmo.

Per farlo è utile tenere a disposizione un paio di esempi in cui, rispettivamente, il funzionamento dell'algoritmo sia a fondamento della domanda del lavoratore o della difesa del datore di lavoro.

Nel primo caso è facile attingere all'esperienza concreta. Pensiamo alla causa in cui il rider chiede l'accertamento della natura subordinata della collaborazione intercorsa con la piattaforma, allegando specifiche modalità di funzionamento del sistema informatico a cui è perennemente collegato durante il lavoro che gli danno direttive puntualissime su come comportarsi, lo controllano in ogni movimento e gli attribuiscono un punteggio che varia di conseguenza e sostenendo che, in tal modo, la piattaforma (la società che gestisce la piattaforma) esercita su di lui il classico potere direttivo, di controllo e sanzionatorio che caratterizza la subordinazione.

In assenza (ancora) di esperienza personale concreta da spendere per il secondo esempio, immaginiamo invece un datore di lavoro con migliaia di dipendenti che abbia utilizzato un algoritmo per operare il licenziamento collettivo di qualche centinaio di essi e affermi di aver fatto esclusiva e corretta applicazione dei criteri di legge.

Come ben sappiamo, per accertare un fatto il giudice ha a disposizione fondamentalmente tre strumenti probatori: i documenti, la prova testimoniale e le presunzioni. In casi particolari può anche ricorrere all'esperimento giudiziale. Per valutare i risultati della prova, ove servano competenze tecniche specialistiche, può quindi avvalersi di un consulente tecnico d'ufficio.

8. La prova documentale: le insidie del codice sorgente e l'utilità dei requisiti

Secondo quanto spiegato nel par. 6, nelle controversie in cui si discute di algoritmo il documento principe è costituito dal codice sorgente.

Il codice sorgente è l'algoritmo.

Avere nel fascicolo il codice sorgente equivale ad avere in aula la persona che ha concretamente realizzato l'attività da accertare (ad es. colui che ha calcolato il punteggio del rider oppure ha stilato la graduatoria dei licenziandi). Quanto spiegato rende molto chiaro, però, che allo stato dell'arte "leggere" direttamente il codice sorgente è una impresa pressoché impossibile nell'ambito di un processo civile e comunque una strada alquanto impervia, in assenza di adeguata documentazione a supporto.

Per fortuna, è poco probabile che il datore di lavoro produca spontaneamente il codice sorgente e ciò per vari motivi: perché sa che vi è la prova che il lavoratore ha ragione, perché non è di sua proprietà o teme che se ne approfitti la concorrenza o, semplicemente perché il suo avvocato non ci ha pensato o non sa come fare esattamente.

Non si può escludere, tuttavia, che un datore di lavoro decida di produrlo, voglia essere davvero collaborativo o solo apparire tale.

Se ciò accadesse, quanto appreso sotto il profilo tecnico rende comunque essenziale nominare subito un CTU, al quale spiegare che cosa dobbiamo accertare e chiedere se ciò che è stato offerto dal datore di lavoro è idoneo e sufficiente a consentire l'accertamento e se, dunque, egli è in grado di portare a compimento l'incarico.

La risposta dipenderà fondamentalmente da due fattori: quanto sia "facile" capire cosa deve fare il software e quanta collaborazione offra il datore di lavoro.

Come si è visto, non esistono standard per stabilire se l'algoritmo è "facile". Dipende certamente dalle dimensioni – che vanno da poche righe a milioni – e anche da come è stato scritto.

La collaborazione richiesta al datore di lavoro che ha utilizzato l'algoritmo consiste nella descrizione chiara e precisa del comportamento del suo algoritmo in base al contesto di interesse e cioè ai dati che esso utilizza e, a seconda della complessità del software, può ricomprendere anche la messa a disposizione degli ambienti di prova e di tutto ciò che serve per far "girare" i singoli pezzi di codice. I limitati mezzi del processo civile, infatti, non sono certo in grado di replicare un tale contesto e dunque, senza la collaborazione dell'azienda, verifiche e simulazioni non sono possibili.

Come sopra evidenziato, non è possibile trovare un CTU che conosce tutti i linguaggi di programmazione esistenti, ma sicuramente un bravo CTU, con

competenze di progettazione e tematiche di ingegneria del software, è in grado di aiutare il giudice nel muoversi tra queste problematiche per focalizzare dove e cosa cercare o chiedere. Per tematiche ed argomenti puntuali, potrà essere necessario un ulteriore supporto di specialisti su determinate materie (es. AI) o linguaggi.

Di certo il CTU dovrebbe avere una laurea in informatica o comunque una forte specializzazione e potrebbe non bastarne uno solo. La ricerca dovrebbe avvenire per passa parola e facendo riferimento a canali esterni al mondo giudiziario, perché allo stato non esiste un "albo" di esperti software.

Anche formulare il quesito non sarebbe facile. Probabilmente converrebbe farsi aiutare dallo stesso CTU, spiegandogli l'obiettivo del suo coinvolgimento e cioè l'oggetto dell'accertamento che deve compiere.

I problemi sinora evidenziati sarebbero gli stessi ove fosse il lavoratore a chiedere l'esibizione del codice sorgente.

In tal caso, tutte le considerazioni che precedono impongono di ponderare bene la decisione istruttoria.

Sembra quanto mai opportuno, ma in realtà è indispensabile, che il giudice approfondisca innanzi tutto le caratteristiche del codice attraverso l'interrogatorio delle parti e che lo faccia con l'assistenza di un CTU.

Solo in tal modo, in particolare con le informazioni tecniche che il datore gli può fornire ove decida di essere collaborativo, è possibile acquisire elementi che consentano di comprendere se sussiste la concreta possibilità di utilizzare il codice ai fini dell'accertamento o se, come già evidenziato, la sua acquisizione rischia di rimanere fine a sé stessa.

Il datore di lavoro potrebbe opporre alla richiesta di esibizione ostacoli legati alla proprietà intellettuale del software ed alla sua segretezza, così impegnando il giudice su un altro fronte giuridico.

Una tale difesa da parte di una pubblica amministrazione dovrebbe essere valutata anche alla luce del diritto di accesso e dei principi di diritto affermati dalla citata giurisprudenza amministrativa in merito all'ostensione dell'algoritmo.

In ambito privato, in cui non esiste un diritto di accesso del lavoratore ai documenti aziendali con cui l'ordine di esibizione possa confrontarsi, tali questioni sono comunque destinate a rimanere prive di grande rilevanza pratica a causa dell'assenza di effettivi strumenti sanzionatori e/o coercitivi per la mancata esibizione e della debolezza dell'unico rimedio a disposizione, costituito dalla possibilità del giudice di trarne meri argomenti di prova.

Le chance di una produzione in giudizio del codice sorgente, dunque, dipendono comunque dalla volontà del datore di lavoro.

Non risulta che finora un codice sorgente sia mai stato prodotto in un giudizio ed è verosimile che ciò non accadrà tanto facilmente.

L'accertamento istruttorio è dunque destinato a svolgersi spesso in altro modo. Grazie a quanto spiegato al par. 6, innanzitutto, il codice sorgente non è l'unico documento esistente.

Ne esiste un altro altrettanto importante costituito dal documento che contiene i requisiti o logiche, cioè le istruzioni che sono state date allo sviluppatore del programma e da costui tradotte nel codice sorgente, utilizzando uno o più linguaggi di programmazione.

L'utilità probatoria dei requisiti è data dal fatto che lo sviluppatore non può modificare le istruzioni e dunque possiamo presumere che il codice sorgente sia conforme alle istruzioni.

L'esistenza di un documento è ovvia, si è visto, essendo di fatto impossibile che le istruzioni di software più o meno complessi siano state impartite solo oralmente.

Un documento da qualche parte deve esistere, insomma.

Il datore di lavoro potrebbe produrlo oppure potrebbe essere il lavoratore a chiederne l'esibizione.

Se la collaborazione del datore di lavoro conduce un tale documento all'interno del fascicolo processuale, l'accertamento istruttorio dei requisiti, cioè di quali sono le istruzioni fornite allo sviluppatore, ci riporta sul terreno più familiare della valenza probatoria del documento in cui essi siano contenuti.

Potrebbe trattarsi di documento avente tutte le caratteristiche per provare da solo il suo contenuto, ad esempio il contratto scritto – firmato e registrato – tra committente e programmatore esterno alla sua organizzazione aziendale che contenga appunto le indicazioni da sviluppare. Se così fosse, potremmo probabilmente fare a meno di ogni altro accertamento.

Non è detto, tuttavia, che le istruzioni abbiano una tale veste documentale: chi le impartisce allo sviluppatore, in quel momento, non si preoccupa di doverle successivamente dimostrare in giudizio e magari potrebbe anche non avere alcun interesse a farlo.

In tal caso, non diversamente da quanto accade ogni volta che viene prodotto un qualche documento che non ha una forza probatoria autonoma, sarà necessario che la sua autenticità venga confermata in sede testimoniale e dunque che il giudice acquisisca la dichiarazione giurata di chi possa affermare che, effettivamente, quelle contenute nel documento sottopostogli sono le istruzioni date allo sviluppatore.

In questi casi il teste ideale è ovviamente il programmatore stesso, il quale dichiari che il programma è il frutto della fedele traduzione da parte sua di quelle istruzioni in linguaggio di programmazione. Per i software di grandi dimensioni, spesso affidati ad un gran numero di sviluppatori che operano in giro per il mondo, ciò potrebbe però rivelarsi irrealizzabile.

Probabilmente potrebbe essere un teste adeguato anche chi ha consegnato allo sviluppatore il documento contenente le istruzioni, le abbia o meno redatte lui, purché sia in grado di riconoscere nel documento sottopostogli quello originale.

In ogni caso, anche la presenza dei requisiti in atti rende quanto mai opportuna l'assistenza di uno o più CTU.

Essi, infatti, sono scritti in linguaggio tecnico e sono tanto più complessi quanto lo è il software. Sarebbe dunque pericoloso per il giudice avventurarsi da solo nella loro analisi ed interpretazione.

Il profilo documentale contiene un'altra insidia.

È intuitivo, infatti, il serio problema per l'accertamento giudiziale del contenuto dell'algoritmo rappresentato dal fatto che il software è soggetto a continui aggiornamenti.

Man mano che passa il tempo, l'algoritmo si discosta sempre più dalle istruzioni originarie. Ottenere la prova di queste ultime, dunque, non è affatto sufficiente. È anche necessario accertare che le istruzioni relative alla parte che interessa non siano state modificate o venire in possesso di quelle successive ad essa relative, "inseguendo" gli atti scritti in cui esse sono contenute.

Il problema sussiste ovviamente anche per il codice sorgente.

Se mai venisse prodotto, dunque, sarebbe essenziale poter verificare che si tratti proprio della versione in uso nel momento a cui si riferisce l'accertamento. Recuperarla non è affatto facile, tuttavia, essendo di fatto possibile soltanto se l'azienda ne conserva traccia nel cd. versioning di cui si è fatto cenno. In entrambi i casi, raggiungere questo risultato è tanto più difficile quanto più complessi sono l'algoritmo e tutto il software.

9. La prova testimoniale e la prova presuntiva

È altamente probabile – si è già detto più volte – che, nella maggior parte dei casi, in atti non sia presente alcun documento idoneo a dimostrare direttamente come funziona l'algoritmo – né il codice sorgente, né i requisiti – e che le parti si affidino interamente alla prova testimoniale.

I limiti di questo mezzo istruttorio sono evidenti.

L'algoritmo sostituisce la persona nelle operazioni che gli sono affidate.

L'unico "soggetto" che potrebbe essere "interrogato" sul contenuto di queste ultime è dunque l'algoritmo stesso, andando a recuperare la sua "memoria" di ciò che interessano il giudizio. Non esistono persone che possano riferire direttamente come esso ha concretamente operato, se non avendo la possibilità e la competenza per andare a "leggere" il codice sorgente.

Al momento, le cause dei rider sono la principale palestra in cui possiamo verificare potenzialità e limiti della prova testimoniale.

In tali cause entrambe le parti articolano capitoli di prova sul funzionamento del software indicando come testi soggetti che non lo hanno sviluppato, né hanno mai visto i requisiti: altri rider e dipendenti con varie funzioni aziendali.

Costoro non sono in grado di dire per esperienza personale come ha funzionato in concreto il software nelle occasioni che interessano la decisione, né come funziona in generale.

Essi possono solo riferire ciò che è stato detto loro da altri oppure ciò che hanno dedotto dalla loro personale esperienza concreta degli effetti dell'operatività del software.

Nel primo caso la loro deposizione appare pressoché inutile. Se il teste non è né il programmatore, né colui che ha redatto e consegnato i requisiti, infatti, quando dichiara che l'algoritmo funziona in un certo modo, offre di fatto mere valutazioni sintetiche che non sa spiegare se non *de relato*, perché non è in grado di aggiungere i dettagli di fatto su cui esse si fondano.

Nella seconda ipotesi il risultato della prova testimoniale ha maggiori chance di essere utile al giudice, in quanto può offrire elementi idonei a fondare un ragionamento presuntivo.

Il teste che ha subìto o gestito i risultati dell'attività svolta dal software, in effetti, può fornire al giudice fatti che gli consentano di giungere in termini presuntivi ad una certa conclusione sulla modalità di funzionamento dell'algoritmo. Se, ad esempio, i testi del rider ricorrente riferiscono in modo convincente – magari anche con l'ausilio di qualche screenshot della loro app – di un certo numero di occasioni in cui, dopo aver omesso di connettersi per qualche giorno, hanno visto che il loro punteggio è sceso, il giudice può presumere che effettivamente l'algoritmo ricollega una decurtazione del punteggio alla mancata connessione. Analogamente è a dirsi per i dipendenti della piattaforma che abbiano accesso ai dati relativi all'attività del singolo rider e ai relativi punteggi e possano affermare di aver concretamente verificato che invece, nei giorni successivi alle mancate connessioni, il punteggio era lo stesso che in precedenza.

È il ragionamento presuntivo, al momento, lo strumento probatorio su cui verosimilmente il giudice può fare maggiormente conto.

Ove venga puntualmente provato per testi e/o documenti – o, eventualmente, non sia contestato – che in un certo ambito si verifica sempre un certo fatto che presuppone una determinata logica, in effetti, si può iniziare a presumere l'esistenza di una regola corrispondente. Così, nell'esempio del licenziamento collettivo attuato con algoritmo, se venisse provato che i dipendenti licenziati erano tutti senza figli, mentre i "salvati" li avevano, si potrebbe forse iniziare a presumere che l'algoritmo ha davvero utilizzato il criterio dei carichi di famiglia.

È tuttavia essenziale che i fatti offerti al giudice per risalire presuntivamente al funzionamento dell'algoritmo siano adeguatamente provati e che il ragionamento presuntivo segua una logica stringente.

Sotto il primo profilo non sembra sufficiente che il teste che aveva accesso ai dati concernenti i rider affermi di aver verificato che il punteggio non variava, essendo necessario che vengano documentati i dati a cui si riferisce ed il giudice possa così verificarli ed analizzarli direttamente.

Sotto il secondo profilo, è necessario che i dati forniti siano tutti quelli relativi al periodo dedotto in giudizio e che, in presenza di una qualsiasi incertezza o incompletezza, il ragionamento non si fermi alla verifica del singolo criterio controverso, ma venga esteso a tutti i criteri che incidono sull'attribuzione del punteggio.

10. I poteri d'ufficio

È probabile che spesso le prove offerte dalle parti non risultino adeguate, mancando sia prove dirette del funzionamento dell'algoritmo (codice sorgente e requisiti), sia la prova di fatti idonei ad integrare indizi gravi, precisi e concordanti del fatto che esso funziona così come affermato dall'una o dall'altra parte. Il problema è lo stesso ove il datore di lavoro produca un codice sorgente di milioni di righe, scaricando sul giudice l'onere di ricercare al suo interno la prova delle sue allegazioni.

In questi casi al giudice non resta che ricorrere agli strumenti messi a sua disposizione dall'ordinamento per risolvere analoghe situazioni: la generale regola sulla ripartizione dell'onere della prova di cui all'art. 2697 c.c. e prima ancora, nel processo del lavoro, l'esercizio dei poteri istruttori d'ufficio.

Ove sussistano i presupposti individuati dalla giurisprudenza perché il giudice possa fare ricorso a questi ultimi[3] e dagli atti o dall'interrogatorio di parte convenuta abbia tratto spunti per individuare il documento che contiene i requisiti e chi può adeguatamente confermarlo in via testimoniale[4], il giudice potrebbe anche disporre d'ufficio l'esibizione di tale documento e la relativa prova testimoniale.

Teoricamente il giudice potrebbe anche ordinare l'esibizione del codice sorgente. Le considerazioni sopra svolte in ordine alla difficoltà di leggere tale

---

[3] E dunque, in particolare, esista già in atti una cd. *semiplena probatio* che ne esclude il carattere meramente esplorativo.

[4] Il che significa che, senza la collaborazione della parte convenuta, di fatto questi poteri d'ufficio non possono essere esercitati.

documento informatico, tuttavia, suggeriscono di essere estremamente prudenti nel varcare una soglia che potrebbe condurre in un vicolo cieco. Meglio dunque evitare di farlo, a meno che gli altri elementi in atti non consentano di accertare l'esistenza di tutte le condizioni già accennate che ne consentono l'utilizzo. Anche in questo caso, sarebbe quanto mai opportuno farsi assistere sin dall'inizio da un CTU che aiuti a compiere una tale valutazione.

Sicuramente il margine di approfondimento istruttorio d'ufficio è più ampio in relazione ai tanti fatti che possono fondare un ragionamento presuntivo. A fronte di prove testimoniali insufficienti sui criteri di attribuzione del punteggio al rider, ad esempio, il giudice potrebbe disporre la produzione dei dati relativi alle specifiche variazioni subite dal punteggio del ricorrente ed alle sue connessioni in un certo periodo e, ove essi vengano contestati dagli interessati, acquisire una prova testimoniale in ordine alla loro genuinità.

11. L'onere della prova

In mancanza di una *semiplena probatio* e/o di una pista probatoria da seguire, al giudice non rimane che applicare la regola sull'onere della prova.

Neanch'essa risulta di facile applicazione in questi casi.

Lo strumento principe per dimostrare come funziona l'algoritmo, infatti, è dato dall'algoritmo stesso il quale, come si è visto, si trova espresso, in linguaggio comune, nei requisiti e, in linguaggio di programmazione, nel codice sorgente. Requisiti e codice sorgente, tuttavia, sono in possesso del solo datore di lavoro. Ove l'applicazione dell'ordinaria regola di cui all'art. 2697 c.c. accolli l'onere probatorio al datore di lavoro, il discorso è agevole.

Se il datore ha deciso di usare algoritmi e poi non riesce a dare la prova a suo carico di come essi funzionano, il fatto che perda la causa rientra nell'ordine delle cose. Aveva l'onere di provare e tutti gli strumenti per farlo: *imputet sibi*. Il discorso diventa problematico, però, quando l'onere della prova grava sul lavoratore.

Sappiamo che l'onere della prova, a volte, è "crudele" – è capitato a tutti di dare torto ad una parte intuendo che aveva ragione, ma non è riuscita a provarlo – e che tale esito è un male inevitabile.

Ciò vale, tuttavia, soltanto quando le parti sono alla pari, quando cioè hanno lo stesso rapporto con ciò che va provato.

Far pagare le conseguenze della mancanza di prova o della sua insufficienza a chi non ha e non ha mai avuto in mano gli strumenti per offrirla, invece, cozza contro la logica e il più elementare senso di giustizia.

Il problema si pone con una certa frequenza nelle cause di lavoro.

Spesso il lavoratore subisce gli effetti di decisioni e vicende aziendali che si concretizzano senza alcun suo coinvolgimento e sono anche al di fuori della sua sfera di conoscibilità, come ad esempio il trasferimento o il licenziamento per GMO oppure i mutamenti organizzativi aziendali da cui derivano cambiamenti di orario o delle modalità lavorative.

In tutti questi casi il lavoratore è in difficoltà a fornire la prova delle vicende che, di fatto, sono a fondamento della sua domanda e un'applicazione rigorosa della regola di cui all'art. 2697 risulta ingiusta e vessatoria.

In alcune situazioni il legislatore, consapevole di ciò, è direttamente intervenuto sull'onere probatorio come, ad esempio, in materia di licenziamento con l'art. 5 l. 604/1966.

In altri casi è la giurisprudenza che accolla al datore di lavoro l'onere della prova, ad esempio delle ragioni tecniche, organizzative e produttive che ai sensi dell'art. 2103 c.c. giustificano il trasferimento o del fatto che l'organico aziendale non integra il requisito dimensionale di cui all'art. 18.

Alcuni di questi orientamenti giurisprudenziali fanno espresso riferimento ad un aspetto che già dal 1800 è oggetto di riflessione giuridica e in alcuni ordinamenti è stato addirittura valorizzato a livello normativo: la vicinanza della prova[5].

Nell'ordinamento italiano essa costituisce ormai l'oggetto di un vero e proprio principio[6] – variamente utilizzato anche nelle controversie di lavoro[7] e ricondotto all'art. 24 Cost. e al divieto di interpretare la legge in modo da rendere impossibile o troppo difficile l'esercizio dell'azione in giudizio[8] – secondo il quale, quando un certo fatto sia esclusivamente (o quasi) nella disponibilità materiale di una parte diversa da quella che ha l'onere di fornirla in base alla regola generale di cui all'art. 2697 c.c., l'onere probatorio e le conseguenze del suo mancato assolvimento vanno posti a carico di chi ha tale disponibilità.

A parere di chi scrive, è proprio il principio di vicinanza (o riferibilità) della prova che, nelle cause in cui la decisione dipende dal funzionamento dell'algoritmo, potrebbe offrire al giudice un serio ausilio per integrare e temperare gli effetti di un'applicazione troppo meccanica e rigida della regola di cui all'art. 2697 c.c.

---

[5] Per un approfondimento del tema si rinvia a "La vicinanza della prova" di Chiara Besso in riv. dir. proc. 2015 p. 1383 e ss.

[6] La sentenza che lo ha consacrato viene individuata in S.U. n. 13533/2001 che ha sancito che l'onere del debitore di provare l'adempimento opera non solo a fronte della domanda del creditore di adempimento, ma anche di risoluzione e di risarcimento del danno.

[7] Si vedano, in relazione al requisito dimensionale, S.U. n. 141/2006 e tutte le successive conformi; in tema di prova dell'incremento della produttività aziendale ai fini della corresponsione di un premio Cass. n. 20484/2008; in tema di discriminazione Cass. n. 5476/2021.

Farne uso nelle controversie in cui si parla di algoritmo, ovviamente, non significa stabilire in modo semplicistico che, in tali cause, qualunque sia l'allegazione in ordine a come esso funziona compiuta dal lavoratore onerato della prova, è il datore di lavoro che deve dimostrare il contrario.

Il ricorso a tale principio, infatti, richiede comunque un'attenta valutazione caso per caso che faccia continuo riferimento alla sua *ratio*, che è quella di esonerare la parte onerata ai sensi dell'art. 2697 c.c. dalle conseguenze della mancanza o insufficienza della prova necessaria quando – e solo quando – essa non è nella sua disponibilità, bensì in quella della controparte.

Ciò implica che la parte onerata della prova di un determinato fatto posto a fondamento della sua difesa è comunque tenuta ad offrire ogni prova dello stesso che sia a sua disposizione e può avvantaggiarsi del suddetto principio soltanto a due condizioni: a) se lo ha fatto e quanto così offerto non risulta sufficiente e b) se l'unica prova che potrebbe colmare l'incertezza istruttoria non è in suo possesso, bensì esclusivamente (o quasi) nella disponibilità materiale dell'altra parte.

È per tale motivo che la giurisprudenza esclude l'applicabilità del principio di vicinanza della prova in ambito di lavoro pubblico in relazione a documenti in possesso della sola amministrazione quando "l'interessato abbia la possibilità, secondo le regole di cui al diritto di accesso agli atti della P.A. o eventualmente sulla base degli strumenti processuali a tal fine predisposti dall'ordinamento, di acquisire la documentazione necessaria a suffragare le proprie ragioni"[8].

Riprendendo l'esempio della controversia promossa dal rider, sembra corretto ritenere che quest'ultimo non può limitarsi ad affermare che l'algoritmo penalizza l'irregolarità della sua connessione (così vanificando la libertà di non connettersi formalmente attribuitagli e brandita dall'azienda per dimostrare la natura autonoma della collaborazione) e a far valere il fatto, ovvio, che l'algoritmo con cui potrebbe dimostrarlo non è in suo possesso per invocare l'onere della piattaforma di dimostrare che, in realtà, non è così.

In questi casi il rider deve comunque illustrare al giudice il ragionamento presuntivo da cui scaturisce la sua affermazione, allegando e provando i fatti che ne sono il fondamento come, ad esempio, il fatto che in ripetute occasioni alla mancanza di connessione per uno o più giorni ha fatto concretamente seguito una riduzione del punteggio.

L'effetto del principio di vicinanza della prova si coglie ove il giudice non dovesse ritenere sufficienti gli elementi offerti dal ricorrente – ad esempio perché la prova del punteggio iniziale, delle sue successive variazioni e delle occasioni in cui non si è connesso riguarda un numero non significativo di occasioni – e

[8] Così Cass. n. 12490/2020

in atti non vi fossero elementi sufficienti per ritenere provata neanche l'opposta tesi di parte convenuta.

La mera applicazione dell'art. 2697 c.c., infatti, condurrebbe a ritenere non provato quel certo fatto.

Facendo applicazione del principio di vicinanza della prova, invece, il giudice può valorizzare il fatto che il ricorrente non è in grado di offrire altro al riguardo e che la prova mancante è data dal codice sorgente e dai requisiti, che sono in possesso della piattaforma, e può dunque porre a carico di quest'ultima le conseguenze dell'incertezza istruttoria, ritenendo provato il fatto allegato dal ricorrente sebbene non compiutamente provato dal medesimo.

Conclusioni

Giunti al termine, si sente l'esigenza di sottolineare che, come precisato all'inizio, la presente relazione ha un carattere meramente introduttivo rispetto ad un tema che impegna il giudice – ogni giudice, non solo quello del lavoro – su un fronte nuovo e complesso.

Per tale motivo, ci vengono richiesti grande lucidità nel realizzare le difficoltà tecniche che esso presenta ed un conseguente particolare sforzo di aggiornamento, anche in discipline estranee alla nostra professionalità.

Soltanto l'esperienza concreta dei prossimi anni consentirà di comprendere se i problemi sollevati in questa relazione siano più o meno effettivi e di mettere a punto soluzioni corrette, utilizzando al meglio gli strumenti offerti dal diritto processuale.

La sfida riguarda tutti: si tratta di scrivere una nuova pagina del diritto processuale del lavoro e tutti sono chiamati a dare il loro contributo.