School of Industrial and Information Engineering
Master of Science in Automation and Control Engineering

POLITECNICO
MILANO 1863

# A SCHEDULING ALGORITHM FOR HUMAN-ROBOT COLLABORATION WITH UNCERTAIN DURATION OF TASKS: A FUZZY APPROACH

Supervisor: Prof. Paolo Rocco
Co-supervisor: Ing. Andrea Casalino

MSc Thesis of:
Eleonora MAZZOCCA
Matr. 872158
Maria Grazia DI GIORGIO
Matr. 878051

Academic Year 2017 - 2018

*Alle nostre famiglie.*

# Acknowledgements

Ringraziamo il professore Paolo Rocco per averci saputo trasmettere l'interesse e la curiosità che ci hanno spinto a cominciare questo lavoro di tesi. Grazie per la fiducia che ha riposto in noi e per i suoi preziosi consigli.

Un ringraziamento particolare va ad Andrea Casalino, per la sua infinita disponibilità e gentilezza. Grazie, perché non ci hai mai fatto mancare il tuo sostegno, chiarendoci dubbi e incoraggiandoci ogni qualvolta ne avessimo bisogno. Lavorare al tuo fianco è stata un'occasione unica a cui diamo un immenso valore: la tua propensione alla sfida e la tua capacità di vedere sempre il bicchiere mezzo pieno, ci hanno instillato la fiducia di cui avevamo bisogno per arrivare a questo giorno e la passione per proseguire da oggi in poi sulle nostre gambe.

Un grazie speciale all'autentico ferrarese Nicola: come ci hai insegnato, un nuovo strumento si aggiunge dopo gli 8/4 di una canzone per stimolare la curiosità dell'ascoltatore. Allo stesso modo, negli ultimi mesi ti sei unito al nostro gruppo di lavoro e con la tua simpatia hai reso anche i momenti più stancanti un piacevole ricordo.

Un sentito grazie ai membri del laboratorio MERLIN che si sono mostrati disponibili ad aiutarci e premurosi nell'assicurarsi che il nostro lavoro venisse svolto nel migliore dei modi.

*Maria Grazia e Eleonora*

Ringrazio i miei genitori per avermi supportato in ogni momento ed essere stati l'esempio da seguire: da voi ho imparato che con volontà e determinazione si può giungere ovunque e che non vi è alcuna scadenza temporale oltre la quale si è giustificati a perdere di vista gli obiettivi preposti.
Soprattutto, vi ringrazio per avermi insegnato che la strada è sempre in salita e che la gratificazione va ricercata anche nei percorsi più tortuosi.
In particolare, grazie a mia madre per la dolcezza e delicatezza con cui mi

è sempre stata accanto, sopportando i miei lati più difficili e al contempo insistendo caparbiamente per fare emergere il meglio di me.
Allo stesso modo ringrazio mio padre che da sempre mi ha trasmesso la passione per la scienza: il tuo immenso sapere e il tuo modo di esternarlo sono grandi fonti di ispirazione per me.

Un ringraziamento speciale è dedicato a mio fratello Danilo, per il sostegno che non mi ha mai fatto mancare e per avermi sempre dimostrato di essere orgoglioso di me. Il modo in cui mi conosci è unico: sai come consigliarmi sapientemente, fornendomi le tue prospettive da adulto e assicurandoti che anche io faccia le mie scelte consapevolmente; allo stesso tempo, sai come ripristinare il rapporto che avevamo in infanzia, riportandoci a vivere i momenti spensierati del passato, trascurando almeno per un po' i chilometri di distanza che ci separano nella vita di tutti i giorni.

Un pensiero va a tutti i miei nonni, che avrebbero voluto essere presenti in questo giorno che da sempre attendevano ed immaginavano con gioia: l'orgoglio con cui a tutti raccontavate dei vostri nipoti e l'entusiasmo che mostravate per ogni piccolo traguardo mi fanno sentire intensamente la gioia con cui stiamo condividendo l'importanza di questo giorno.

Vorrei ringraziare le mie zie, Anna e Laura, per aver condiviso con me anche a distanza la gioia dei miei successi.
Un grazie speciale anche a mia zia Rita, per essersi sempre assicurata che tutto procedesse per il meglio, e a mio zio Salvatore, che sono certa avrebbe gioito di questo giorno importante.

A tutti gli amici conosciuti a Milano voglio rivolgere un ringraziamento particolare: di ognuno porto ricordi unici e indelebili.
Un profondo ringraziamento a chi mi ha dedicato tempo e attenzioni, aiutandomi a mantenere chiaro il mio obiettivo e condividendo con gioia i momenti felici e con parole rincuoranti le preoccupazioni; a chi è stato per me una guida, in grado di farmi vedere il disegno più grande, ciò che viene dopo e al di fuori dell'Università, accompagnandomi fino alla conclusione di questo percorso e dando importanza a ogni singolo momento di esso.
Grazie a chi si è preoccupato che il lavoro potesse essere svolto nel migliore dei modi, leggendo e rileggendo la tesi per fornire consigli sinceri.

Ringrazio le mie coinquiline, con cui ho avuto il piacere di condividere la mia quotidianità. Una nota speciale è riservata a Valeria (o Giovi) che fin dall'inizio ha reso divertente e indimenticabile la permanenza nella

nostra casa: in questo periodo siamo cresciute un po' assieme, l'equilibrio e la sintonia che siamo riuscite a creare sono stati essenziali per affrontare le esperienze di questi anni. Grazie per esserti presa cura di me, soprattutto nei momenti in cui ne ho avuto veramente bisogno.

Ringrazio anche chi ormai c'è da tanto tempo: Alessandro che riesce ad annientare la distanza, dimostrandosi sempre presente. Dagli anni in cui condividevamo i banchi di scuola nulla è cambiato, se non che, forse, ora siamo un po' più maturi. Ti sono grata per tutto il sostegno che hai sempre mostrato nei miei confronti, per essere il primo medico che contatto ad ogni mia paranoia e, in generale, per essere fonte di consigli e parole di conforto, in ogni situazione. Buona parte del raggiungimento di questo obiettivo la devo a te che ogni estate riesci a trovare "IL" concerto da non poter perdere, immancabilmente in sessione di esami e, per fortuna, immancabilmente di buon auspicio. Senza le tue stimolanti iniziative non sarebbe stato così divertente.

Infine, grazie alla mia compagna di tesi per avermi accompagnato in questo percorso; l'esperienza condivisa con te ha un valore che non riuscirò a descrivere in queste poche righe. Il modo casuale in cui ci siamo conosciute, unito al modo in cui assieme abbiamo affrontato tutte le vicende che si sono presentate lungo il percorso, mi fa convincere che in realtà niente è stato affidato al fato. In poco tempo abbiamo instaurato un rapporto che va al di là dell'essere semplicemente colleghe universitarie; da te ho appreso molto: mi hai mostrato come guardare il lato positivo delle cose e apprezzare ciò che delle persone si vede solo se vi si dedica tempo ed attenzione. In te ho visto una forza unica: in ogni situazione di sconforto non mi hai mai concesso di demoralizzarmi, anche quando tu per prima avevi bisogno di essere rassicurata.
Grazie per avermi chiesto ogni giorno: "Ma Ele tutto a posto?"; per quanto il tuo ripeterlo spesso ed insistentemente potesse a volte irritarmi, la verità è che ormai è quel momento della giornata che aspetto, anche solo per risponderti con un conciso "Sì": è la conferma quotidiana che per me ci sei sempre.
Custodirò con cura tutti i nostri ricordi, a partire dai primi esami preparati insieme, passando per i momenti spensierati e di svago, fino agli ultimi in cui la stanchezza ha sempre fatto da protagonista: le notti insonni perché il tempo per studiare non era abbastanza, l'aver reso un'abitudine lo spegnere le luci del Dipartimento perché eravamo sempre le ultime a lasciarlo e, ancora, la volta in cui vi siamo rimaste chiuse perché non avevamo realizzato quanto tardi fosse.

Sono tutte cose che - seppur abbiano richiesto sacrifici e forza di volontà - sceglierei di rifare, insieme.
Congratulazioni a noi!

*Eleonora*

Il grazie che rivolgo ai miei genitori, viene dalla più intima parte di me, da quella parte così astratta e così viva da cui sento provenire il sorriso dei ricordi e le lacrime della nostalgia. Ogni mio passo avanti, quando lontano da voi, l'ho fatto con la consapevolezza che radici enormi, forti e sicure mi avrebbero dato sostentamento e che qualunque vento o tempesta avrebbe solo rinvigorito la mia chioma.
E' a voi che dedico questo mio successo, che è anche il vostro.

Grazie a mio fratello Giovanni, sempre presente nonostante i chilometri che ci separano. Sei per me un supporto sicuro, un consigliere sincero, un amico senza riserve. Nel guardaroba della natura c'è un mucchio di costumi e quello di fratello e sorella ci calza a pennello; lo indosso con orgoglio e so che non si consumerà mai.

Un ringraziamento particolare a Francesca: sei stata la prima persona su cui poter contare quando sono arrivata a Milano. La nostra casa ha nel mio animo l'immagine dolce delle cose rassicuranti e sempre ti avrò nel cuore come una complice compagna di viaggio.

Ringrazio i miei zii Raffaela, Walter, Giovanna e Nino, per essere stati un conforto e un punto di riferimento. Stare con voi è rigenerante e il pensiero di avervi vicini è, e sarà sempre, profondamente rassicurante.

Nonna Bettina, non so esprimere a parole la felicità di averti qui oggi. Grazie, dell'amore dolce che mi trasmetti.

A nonna Maria Grazia e nonno Giovanni, grazie per non avermi mai fatto mancare il vostro supporto anche da lontano. La mia gioia per ogni successo conseguito è sempre stata anche la vostra gioia; ogni ostacolo, una scusa per ricordarmi quanto valgo e quanto più grande sarebbe stata la nostra soddisfazione una volta superato.

Ai miei zii Gabriele e Sandra, per l'affetto che non mi avete mai fatto mancare e per essere sempre stati orgogliosi di me, grazie.

Grazie ai miei carissimi amici, fonte inesauribile di sostegno e amore. Mi sento grata, perché so che la mia felicità è anche la vostra.

Ringrazio chi in questi anni mi ha dato fiducia, chi mi ha dedicato tempo, chi mi ha lasciato qualcosa di sé. A quanti hanno contribuito a dare a questa esperienza un inestimabile valore, grazie infinite.

Infine Eleonora, preziosissima amica, “se non ci fossi stata tu, questo affare non si sarebbe concluso!”.
Ti ringrazio di cuore, per aver dato alla parte più pessimista di me la dimostrazione che vale la pena partire, con un po’ di fiducia nel cuore, perché qualcosa di bello succede, come l’averti incontrata.
Questa tesi è stata la nostra Itaca! Sempre l’abbiamo avuta in mente, raggiungerla è stato il nostro pensiero costante. Oggi mettiamo piede sull’isola, ricche dei tesori accumulati per strada.
La tua amicizia è per me tra i tesori più preziosi e Itaca solo una delle infinite mete che raggiungeremo!
Auguri dottoressa Ele!

*Maria Grazia*

# Contents

# List of Figures

# List of Tables

# Sommario

Negli ultimi anni, uno dei cambiamenti più importanti a cui si assiste in ambito industriale e manufatturiero è l'evoluzione verso una robotica di tipo collaborativo, in cui il robot interagisce con l'umano per assisterlo nell'esecuzione di azioni che ne mettono a rischio l'incolumità o che richiedono elevata accuratezza. Affinché la collaborazione dei due agenti sia proficua e porti ad un miglioramento dell'efficienza del processo produttivo, occorre fare in modo che gli agenti coinvolti nel processo di assemblaggio siano capaci di sincronizzarsi. A tal fine, risulta importante predire le azioni dell'operatore umano per controllare il robot in maniera sicura e funzionale. In ambito collaborativo le interazioni tra agenti possono avere risvolti imprevedibili: il robot potrebbe essere costretto a fermarsi per evitare situazioni pericolose per l'umano o, ancora, l'umano potrebbe impiegare tempi variabili per completare una specifica azione. Tale incertezza nella durata delle azioni degli agenti è il principale problema affrontato in questa tesi. Al fine di controllare le azioni svolte nel tempo dal robot, in modo da minimizzare il tempo di inattività degli agenti e massimizzare la produzione, è stato sviluppato un algoritmo di scheduling che, tramite l'impiego della Fuzzy Theory, è in grado di rappresentare le durate delle azioni degli agenti tramite distribuzioni possibilistiche. Questo consente di implementare un approccio diverso dai classici metodi finora proposti. I risultati sperimentali, ottenuti in uno scenario reale di assemblaggio collaborativo, indicano che l'algoritmo sviluppato garantisce efficienza nel comandare il robot, il quale si adatta adeguatamente alle esigenze del collaboratore umano, assecondando prontamente le sue intenzioni di collaborazione e minimizzando i suoi tempi di inattività.

# Abstract

One of the most important changes in the manufacturing industry is the evolution towards collaborative industrial robotics, in which robots interact with a human to carry out actions that might be dangerous for the human's safety or that require high accuracy.
To achieve an efficient collaboration as well as an improvement in the production process, it is necessary to ensure that the involved agents are able to mutually synchronize.
To this aim, it is important to predict the actions of the human operator and consequently control the robotic actions in a functional way.
In the collaborative context, interactions between agents can have unpredictable consequences: the robot could be forced to stop, to avoid dangerous situations for the human, or again, the human could spend a variable amount of time to complete a specific action.
This uncertainty in the duration of actions of the agents is the main problem faced in this thesis. In order to minimize the inactivity time of agents and maximize the production, a scheduling algorithm has been developed, based on Fuzzy Theory, which is used to represent the durations of agents' actions through possibility distributions.
The experimental results, obtained in a real scenario of collaborative assembly, show that the algorithm developed guarantees an efficient control of the robot, which adequately adapts itself to the needs of the human collaborator, promptly satisfying his or her intentions and minimizing his or her inactivity time.

# Chapter 1

# Introduction

## 1.1 Human-Robot Collaboration and Industry 4.0

After many years of conventional procedures of production, industrial manufacturing is evolving toward flexible and intelligent manufacturing, the so-called Industry 4.0. One of the key elements of such revolution is the interaction of human and smart machines, which allows to achieve the greatest flexibility and productivity simultaneously.
A particular attention is oriented to collaborative manufacturing systems, where a human worker cooperates close to collaborative robots, in production scenarios.
This evolution means breaking with the established practice of separated workspaces between robot and human. These changes are reflected in new safety standards related to collaborative robotics. Indeed, a wide field of research focusing on the prevention of human-robot impacts and/or the minimization of related risks or their consequences arose. Therefore, an important issue in human robot collaboration is to enhance safety through the implementation of collision avoidance systems. Several methods for the estimation of the proximity from a robot to an object and the generation of alternative trajectories and acceleration/velocity variations before collision have been developed, used for strategies aimed at preventing undesirable robot-human impacts when they are working together.
Therefore, since the collaboration must ensure the safety of the human operator, the robot might be forced to slow down its movements or even stop during the execution of an action.
These unexpected changes in the robot trajectories lead to a very imprecise knowledge regarding the duration of its actions. In addition, a big source

of uncertainty is represented also by the cooperating humans, whose tasks may have considerably variable durations.
Therefore, the problem of managing temporal flexible events plays an important role when developing a performing human-robot collaboration. Indeed, the actual problem is to schedule the cooperative tasks in real time, managing the complexity arising when having a great number of cooperating agents, sharing areas and production steps.
Thus, a good dynamic task planning framework is needed to coordinate tasks of robot and human, accounting for the uncertainty affecting the knowledge related to these agents.
The main capability of the planning with uncertainty approach is to control the robot in order to guarantee safety and fluency for the operations of the human. In other words, the goal is to control the robot in such a way to dynamically adapt to the actual behavior of the human.

## 1.2 Thesis contributions

The main objective of this work is to efficiently control the robot, in order to adapt to the needs of a collaborating human.
The collaboration in this work has been intended as cooperation of the agents, aimed at assembling products of common use: some actions are performed by the human and others, for example those for which high accuracy is required, by the robot.
Most of the actions performed by one agent depend on other actions carried out by the other agent. For example, the human might have to assemble two pieces, where one of them is the outcome of a previous assembly performed by the robot.
This leads to a chain of operations that depend one on the other: if the tasks are not optimally coordinated, an agent might be in an idle condition for long time, reducing drastically the overall performance.
The aim of this thesis is to successfully control the robot, scheduling its actions in order to promptly satisfy the human's requests, minimizing the time for which he/she remains in an idle condition and accounting for the uncertainty of the overall system.
In this work, an algorithm has been developed to optimally decide the activities that the robot has to perform, taking into account the imprecise knowledge about the duration of actions of the agents.
The approach used in this thesis to describe the imprecision and uncertainties in the durations of batch processing tasks relates to Fuzzy Set Theory and Possibility Theory. Such frameworks provid methods to represent the

available information related to the agents involved in the system, assigning a degree of “reliability”: thus, the approach quantifies the uncertainty affecting the considered system, allowing to counteract it. The scheduler obtained is then capable of handling significantly uncertain environments, providing good performance and adapting to the human’s needs.
As an additional target of the thesis, the method proposed has been validated experimentally in a realistic collaborative assembly: it has been proven that a reduction in the waiting time of the agents has been achieved, as well as an increase in the productivity.

## 1.3 Organization of the thesis

In the following an overview on the structure of the thesis is given.
Chapter 2 presents the current State of the Art, showing how problems concerning scheduling have been tackled during the years. In particular, an introduction on the fuzzy approach for scheduling is provided, as it constitutes the basis for the algorithm developed in this work.
Chapter 3 presents the method used to model the system considered in this thesis, adapting Fuzzy Time Petri nets (FTPN), which have been chosen amongst other options presented in the same Chapter. In particular, the convenience in choosing such representation of the system is pointed out: the FTPNs allow to account for uncertainty, including it in the time distributions that are associated to the transitions of the net.
A Section is dedicated to the computation of the reachability tree of FTPN, which is a graph illustrating all the possible ways in which the system can evolve, starting from a particular condition. It is used by the scheduler to solve the decision-making problem: analyzing the tree, the scheduler is able to identify the most convenient evolution of the system within a temporal horizon, in order to guarantee an optimal synchronization of the agents.
In Chapter 4 the problem of uncertainty is detailed, starting from an overview on the origins of Possibility Theory. After some theoretical notions, it is presented how the uncertainty affecting the system is modeled and propagated through the reachability tree. The proposed method is compared with the more classical approach based on Probability Theory, justifying the choices made throughout this work.
Chapter 5 shows how the theory ted in Chapter 4 has been applied to the problem treated in this thesis. In particular, it is reported how the durations of actions of the agents are represented using fuzzy numbers. Regarding the operator’s actions, an external predictive algorithm was

exploited to estimate their durations and is briefly presented in the same Chapter.
The main goal of Chapter 5 is to explain the working principle of the scheduling algorithm, starting from the generation of the reachability tree up to its analysis, performed to identify the best evolution of the system. Chapter 5 ends with a pseudo-code that summarizes the implemented scheduling algorithm.
Chapter 6 presents some use cases that has been employed to test the scheduling algorithm: simulations and experimental tests are reported to prove that the scheduler is able to adapt to different situations and conditions, demonstrating that a good performance is provided both in terms of minimization of waiting times and in terms of computational cost.
Chapter 7 concludes this thesis highlighting the results obtained and providing hints about possible future developments.

# Chapter 2

# State of the Art

In real world applications, in which the robot performs a human assistance function, to ensure perfect interaction between the agents it is extremely important to schedule an action plan for the robot that provides adequate assistance at the right time, in order to minimize the waiting time between the two agents.

First of all, an efficient planning of the sequence of actions to be performed by the robot can be done only if it is assumed that the collaborative system under consideration is dynamic, i.e. a system whose evolution is determined not only by the completion of a robotic action, but also from the fulfilment of an action carried out by a human. In many applications, the environment is considered to be static and that it only changes when the robot performs an action, i.e. there are no exogenous variations from the robot point of view [1]. Nevertheless, in real world applications, changes are caused not only by robotic actions but also by exogenous events related to human activities, which then need to be taken into account.
Moreover, the exogenous events from human activities cannot be treated in a simple way, as by considering a "timetable" of human activities, because humans act based on their intentions and perceptions; a prediction on human activities is required to account for them with more accuracy [1].

On the other side, exogenous events from human activities are not perfectly predictable, because human intentions are not fully observable: thus, such uncertainty must be considered during a scheduling stage.
Analogously, robotic actions performed in the real world have a certain degree of uncertainty, because the duration of an activity involved in the process may be difficult to predict accurately: uncertainty concerning what or how much work must be performed to complete an activity leads to a

significant effort in estimating the distribution of activity duration.

To be efficient, scheduling methods need to handle problems where events are related to uncertain durations [1].
To this aim, many algorithm have been proposed during the years, as alternatives to the classic scheduling methods: the uncertain factors in real-world scheduling problems are not easily treated by classical deterministic or stochastic solution approaches.

Deterministic scheduling models [2, 3, 4, 5] assume that all project parameters can be specified in advance, before scheduling [6], but Janak et al. [7] demonstrated that even if it is often assumed that all system parameters are deterministic in nature, even slight variations in system conditions can make a scheduler unfeasible [9].
Stochastic scheduling models [2, 10, 11] consider the project parameters as independent random variables with given distributions, determined according to Probability Theory [6].
So, the uncertainty is viewed as random variable or a stochastic process, involving Markov decision processes (MDPs) and their extensions [12, 13], as well as partially observable Markov decision processes (POMDP) [14, 15, 16, 17, 18, 19, 20, 21].

Nevertheless, there are applications in which collecting enough data to obtain a reliable distribution of the random variable is complicated [6].
Moreover, these techniques are based on Markov processes, in which the future states of a process depend solely on the present state, not on the sequence of events that precede it.
According to this assumption, the time between any two events is represented as an exponential distribution, which in general is not able to provide realistical descriptions, due to its “ memoryless ” property.

Semi-Markov versions of MDPs and POMDPs are then used to counteract such restrictive property: a semi-Markov process is an actual stochastic process that evolves over time. It is a generalization of Markov models that allows the time spent in a particular state to be represented by an arbitrary probability distribution, instead of only by the exponential one, and allows the state transition probabilities to be dependent on the time spent in that state. So, in other words, the process is able to “ remember ” [1, 22] its past.
However, although they are widely used in human activity recognition [23, 24], semi-Markov MDPs and POMDPs models are usually computa-

tionally too expensive and theoretically too complex because of the need to use complicated multiple integration techniques when the uncertainty is represented by continuous distributions [25]. Therefore, it is difficult to apply them for solving practical scheduling problems and alternative treatments of uncertainty have been considered [26].

Several methodologies exist for the explicit consideration of different types of uncertainty within a scheduling model: two-stage stochastic programming, parametric programming, fuzzy programming, chance constraint programming and robust optimization techniques represent different frameworks to take into account parameter uncertainty [9].
In fact, more importance has been given to the uncertainty associated with the system parameters, that can be characterized in three different ways:

- If there is not enough information to construct an accurate estimation of the uncertain parameter's distribution, then such distribution is often represented as an uncertain parameter that is assumed to take values within a specified range, defined by an upper and lower bound [9].

- If there is sufficient information to construct a reliable distribution for the parameter, then the uncertain parameter can be represented in a probabilistic way [9], similarly to the work presented by Petkov and Maranas [27] and Janak et al. [7].

- An alternative to the bounded and known distribution cases is the use of fuzzy sets [9] as noted by Li and Ierapetritou [29].

According to the specific kind of uncertainty, an appropriate modeling approach has to be used.
Thompson and Zawack [28] and Ryu and Pistikopoulos [32] developed a framework to model uncertainty with parametric programming applied to scheduling problems. Li and Ierapetritou [30, 31] used multiparametric programming to take into account multiple forms of parameter uncertainty. Parametric programming handles optimization problems by means of iterative approaches: the required computational time might be a significant limit, especially for large-scale industrial problems [9].
An alternative is offered by different approaches, such as chance constraint programming, robust optimization techniques, and fuzzy programming. The first has been developed by Charnes and Cooper [34, 35, 36], who

removed the constraints on uncertain parameters, to substitute them with their respective probabilistic forms, which explicitly take into account the stochastic nature of the uncertain parameters [9]. Chance programming has been included in scheduling problems by Orcun et al. [37] and Petkov and Maranas [27].
Robust optimization techniques [7, 8], [38, 39, 40, 41, 42, 43, 44, 45, 46, 47], have been developed to guarantee robustness with regard to multiple forms of uncertainty present within the system under investigation. Lin et al. [8] and Janak et al. [7] developed the robust optimization theory for models in which distributions are known and bounded.
When no information about the considered distribution is available, Fuzzy set Theory can be used to model uncertain parameters: fuzzy programming methods treat these uncertain parameters as fuzzy numbers, while their constraints are represented by fuzzy sets [9].
These aspects were theoretically treated by Chanas and Kamburowski [48] and adapted to scheduling problems by Hapke and Slowinski [49] and Wang [50], who made use of Possibility Theory [65].
Wang in [6] applied fuzzy programming to the problem of scheduling under uncertainty, developing a scheduling methodology based on fuzzy set theory for uncertain product development projects, and in his method the imprecise temporal parameters related to the project are represented by fuzzy sets.

The target of this thesis is the implementation of a scheduling methodology based on Fuzzy set Theory and on Possibility Theory, to control a robot in a HRC context, accounting for the uncertainty in the duration of actions.

# Chapter 3

# Human-robot collaborative context

In a Human-Robot Collaboration (HRC) context, the human and the robot cooperate to reach a common goal, that could be for instance the assembly of pieces in a production line.
To this aim, these two agents perform actions that can be independent or cooperative. However, the action of an agent often depends on the success of the action of the other one.
For example, if the robot has to put an object in a box, the box has to first be assembled by the human; otherwise, the robot would be forced to wait for the human to complete such action.
In a HRC context, planning plays a crucial role. This is particularly evident for industrial environments, where the aim is to optimize the production time, in order to maximize the production rate.
An example of optimization could be commanding the robot to perform intermediate actions, while it waits for the human to assemble the box; in this way, the robot would avoid to be inactive for too long.
More formally, task planning consists in the choice of actions to command to the " controllable " agents of the system, e.g. robots. Such choice is based on the prediction of actions to be performed by the " uncontrollable " agents of the system.
In this thesis, the human is assumed to be an external agent and therefore uncontrollable.

The purpose of this Chapter is to introduce the representation used to model the system considered for the work of the thesis. A good solution has been found on Petri nets (PN) (Section 3.1): they allow to illustrate the structure of the system, as well as its evolution.

Moreover, it is easy to compute a reachability tree of the system state, exploiting PNs (Section 3.2). Such tree is a graphical representation of all the possible ways in which the system can evolve, starting from an initial state. Every branch of the tree represents a possible trajectory of the system, in terms of sequence of actions performed by agents in the process modeled through the PN.
Analyzing each branch of the tree, the features of the related path are obtained in order to verify whether these features match the specifications imposed by the user of the application. For example, it is possible to compute the time for which the human would have been forced to wait for every path. If the waiting time for a branch is too high with respect to the user's specifications, that branch is discarded.

As it will be later shown, the purpose of the thesis is to create a scheduling algorithm which analyzes the reachability tree in order to find the most convenient evolution of the system - depending on the specifications required by the application - to then properly command the controllable agents.
Considering for instance a HRC context, it could be required to minimize the time in which the human is inactive. As a consequence, the scheduler should avoid to let the robot be busy for a long period of time if the human is forecast to require the help of the robot in short time.
Concerning the decision-making, with the proposed scheduler the user can specify his/her own criteria for choosing the best plan (i.e. path in the reachability tree, for more details see Section 5.3.2). Such choice is lately converted into command to send to the controllable agent.

In Section 3.1 of this Chapter the notion of Petri Net is presented and justified as choice to represent the system considered in this thesis. Section 3.2 presents the notion of reachability tree and the way the latter is exploited by the scheduling algorithm created in this thesis. In Section 3.3 the system specifications used in the thesis are pointed out.

## 3.1 System modeled through Petri Nets

### 3.1.1 Petri Nets: a general overview

Petri Nets (PN), proposed in [52] by Carl Adam Petri, are a tool for modeling communication and interaction between parallel processes.
In the description of processes it is often needed to represent sub-processes

or activities that can be performed simultaneously, in parallel with each other, and that can depend one on the other. Indeed, it could happen that a certain phase of the process cannot occur until other phases or activities are completed or until certain conditions are satisfied.

PN allow to represent and describe a process globally, and also allow to follow its evolution, permitting to see the state in which the network is located in a certain instant. A Petri net is a graph consisting of “ places ” and “ transitions ”, connected through “ arcs ”.
More precisely, a Petri Net is an oriented bipartite graph, defined by the following 4-pla:

$$N = (\, P, T, Pre, Post \,)$$

where:
$P$: is the set of m places (represented by circles)
$T$: is the set of n transitions (represented by bars)
$Pre$: $P \times T \mapsto \mathbb{N}$ is the pre-incidence function
$Post$: $P \times T \mapsto \mathbb{N}$ is the post-incidence function

The “ state ” of the Petri Net indicates the configuration of the process at a given moment and is represented by a “ marking ”, which describes the location of tokens in the places of the net.
The tokens indicate the progress of the operations described in the process steps: when the system evolves towards other states, tokens evolve towards places.
Places and transitions are connected through arcs, which can in turn have a weight assigned. Transitions represent the activities that can be carried out: the conditions to be able to perform the considered activity are represented by the number of tokens in the places that are “input places” for the transition, i.e. places connected to the transition through arcs going from the places to the transition.
If the number of tokens is greater or at least equal to the weight associated to the related arc, then the operation related to the transition can be performed, which in turn means that the transition can “fire”.
After a firing, the tokens redistribute in the Petri Net, going into the “output” places of the transition, i.e. places connected to the transition through arcs going from the transition to the places; in short, transitions are responsible for the flow of tokens throughout the net.
PN can be used as a “ graphical tool ” to represent and simulate the dynamics and the competing activities present in the system [53].

### 3.1.2 Time Petri net

When dealing with scheduling processes, it becomes necessary to wedge in many events or actions that would happen subsequently.
This introduces the need of modelling the sequence of actions with a certain degree of precision.
To this aim, duration of the involved actions must be considered and therefore PNs that account for time must be used: the evolution of events must be coherent with the evolution of the real system during time.
Many timed extensions have been proposed for PN, which can be roughly divided into two main categories [54]:

- **Timed PN**: they include a " duration " associated with each transition. It is assumed that the duration related to events is perfectly known and so correspond to sharp numbers.

- **Time PN**: they consider an entire " time interval " associated to the relative firing time of each transition. They are used for systems that may evolve in a non-deterministic way.
  Therefore, to every transition a time interval $[\,t_{min}, t_{max}\,]$ is assigned, where $t_{min}$ is the minimal time taken to fire, from the enabling time, while $t_{max}$ is the maximum time required to fire from the enabling time.

- **Stochastic PN**: they are Time PN in which every transition is characterized by a firing delay, which is a stochastic variable with exponential distribution. Hence, the Stochastic PN (SPN) represents a Markovian stochastic process, where the memoryless property is assumed to hold [55]. As a consequence, the transitions that remain enabled after a change in the marking don't modify their firing times, which is independent from the time elapsed since the enabling instant. Assuming that a real system satisfies the memoryless property would be a too strict assumption: SPNs are therefore discarded.

Time PN are the more general model, suitable to depict real time stochastic systems. In fact, in real systems it might be necessary to model transitions from one state to another not happening in a precise time instant. It is possible to assign a time interval accounting for the uncertain evolution of the system.

In HRC contexts, a robot that is performing an action might be forced to interrupt it, if a possible collision with another agent is detected.
Or, again, if a robot is performing an action that depends on the success

of the action of another agent, it might have to remain idled if the other agent is not done yet.
Since it is not possible to know precisely how long the other agent would take to complete the action [56], it is not possible to precisely predict the robot waiting time either and so the duration of the robot action (see Section 6.2.1 for further details).
This unknown aspect of time goes under the name of " uncertainty ".

Hence, it is necessary to include the uncertainty in the firing time of transitions of the Time PN representing the system. In this way, the evolution of the system over time would be modeled in a more realistic way.
A more precise model of the collaborative process would certainly lead to a more optimal scheduling: it would ease the scheduler to find the best sequence of actions to command to the robot, satisfying the system specifications and accounting for the above-mentioned uncertainty.

For this reason, the work developed in the thesis finds its basis in Time PN (from now on TPN). Moreover, the modeling procedure proposed in this Chapter will be extended - as shown in the following Chapters - in such a way that the uncertainty won't be represented simply as an interval or through an exponential distribution. Indeed, considering an interval for which every time value is equiprobable would be reductive. Instead, the purpose is to treat the uncertainty with a proper distribution over time.

A *Time Petri net* is a six-tuple [54]

$$N = (P, T, A, w, M_0, I)$$

where:

- P is the finite set of places, $P = p_1, p_2, \ldots, p_n$
- T is the finite set of transitions, $T = t_1, t_2, \ldots, t_m$
- $A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs from (input) places to transitions and from transitions to (output) places
- $w : A \rightarrow \{1, 2, 3, \ldots\}$ is the weight function of the arcs and every weight is a positive integer greater than 0
- $M_0$ is a marking, defined as a mapping $M : P \rightarrow N^+$, representing the initial state of the net. It is a row vector with $|P|$ elements

- $I : T \to \{\mathbb{R}^+, \mathbb{R}^+\} \cup \{\infty\}$ associates with each transition $t$ an interval $[t_{min}, t_{max}]$

A transition is said to be “ enabled ” when the number of tokens in all of its input places is greater or equal to the weight of the associated arc.

When a TPN is used to model a system, the transitions are associated to actions of the agents (e.g. “ the robot works a piece ”) or to evolution of the system from one state to another (e.g. from “ the robot is busy ” to “ the robot is free ”). The places of the net are associated to states of the system or to its resources (e.g. the robot or its arms, the human, buffers, etc.).
Suppose for instance to model a system in which a human and a robot cooperate to assembly a product. The two agents perform individual actions, but a particular action of the human is subsequent to a specific action of the robot, i.e. the human can perform that action only when the robot has completed the other one.
The sequence of actions to assembly the product is:

1. Action A: the robot takes a piece from the buffer and machines it
2. The robot gives the finished piece to the human
3. Action C: the human works the piece
4. Action B: robot takes the piece worked by the human and puts it on an output pallet

The aforementioned assembly can be modelled by the TPN depicted in Figure 3.1.

Being the main goal of the thesis to account for uncertainty in duration of actions, a method to account for such imprecise information is needed. TPNs are a good basis to such problem, because are able to represent the duration of actions through the time associated to their transitions.
A further addition to this, is to associate the uncertainty to such durations: to do so, the time of transitions is not represented as a sharp number but as a distribution. In this way, the duration of an action is represented as a set of possible values, each of which is characterized by a membership degree (see Chapter 4).
The distributions used go under the name of “fuzzy numbers”: from here on, this kind of TPN will be referred to as Fuzzy Time Petri net (FTPN). Further details on Fuzzy Theory and fuzzy numbers are given in Chapter 4.

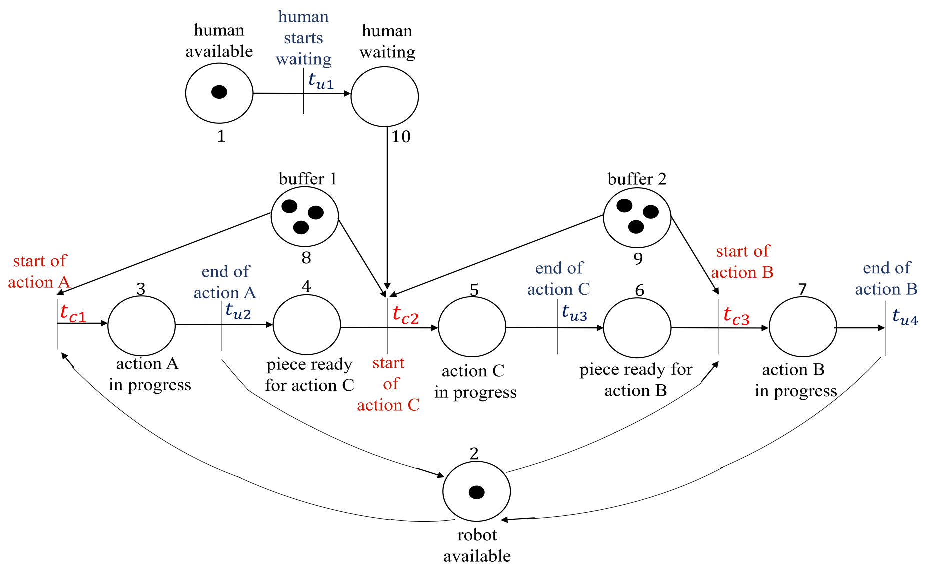


**Figure 3.1:** TPN: example of human-robot collaboration

## 3.2 Decision-making as analysis of the reachability tree of a FTPN

The scheduling algorithm presented in this thesis, actuates a choice about the actions to send to robots. Such actions should be the outcome of a compromise to minimize the waiting time of all the agents involved in the cooperation.
This compromise is found by analyzing the reachability tree of the modelling FTPN, which is computed starting from the current state of the system and developed until a fixed temporal horizon.

The reachability tree of a FTPN is a graphical arrival representation of all the possible reachable markings, with admissible time, starting from an initial marking that describes the starting configuration of the FTPN.

- The nodes of the tree are linked to the reachable markings of the FTPN
- The arcs of the tree are associated to the transitions that bring from a marking of the FTPN to another

An example of reachability tree is shown in Figure 3.2, related to the FTPN of Figure 3.1. Once the reachability tree has been computed, the scheduler

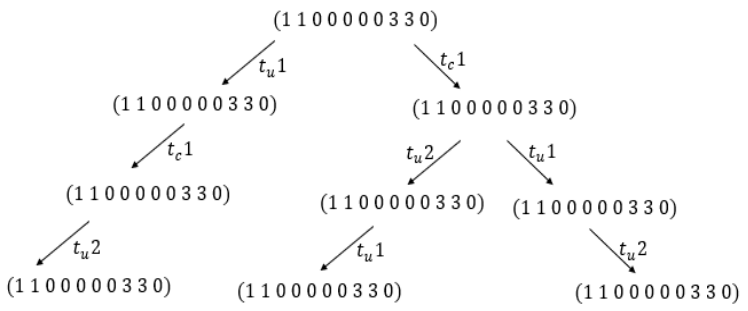


**Figure 3.2:** Reachability tree of the FTPN of Figure 3.1

analyzes every branch, assigning a “ weight ” quantifying the goodness of a path, that depends on some tunable specifications assigned by the user (see Section 5.3.2).
Once the best branch (path) has been chosen, only the first controllable actions in the optimal path are sent to the robot. Then, the algorithm is invoked again and a new tree is computed, to choose the next action to be performed by the robot.
The approach of generating a tree up to a certain temporal horizon to then send to the robot only the first action of the chosen sequence, reflects an MPC approach (see Section 5.3.1).
As it will be proved throughout the thesis, the FTPN combined with the inclusion of the uncertainty (modeled with the method presented in Chapter 4), leads to obtain a reachability tree considerably reduced in size. Indeed, the way the uncertainty is modeled allows to detect the unfeasibility of some branches and therefore their consequent removal from the tree.
The approach anticipated up to now will be thoroughly presented in Chapter 5.

## 3.3 Scheduling preferences

In the HRC context considered in this thesis, the agents considered are humans and robots performing individual or cooperative actions.
Therefore, it is necessary to classify the different classes of actions that will be modeled. They are distinguished according to the agent executing the action and the role having that action, i.e. if it is necessary for other agents to then perform subsequent actions.
Hence, the actions can be categorized as follows [73]:

- **Robot action**: is an action executed autonomously by the robot, which is not directly necessary to satisfy a human's need.
- **Robot to Human action**: is an action performed exclusively by the robot but its outcome influences an action that has to be performed by the human immediately after. Therefore, since the target is to minimize the waiting time of the agents, it is important to give precedence to this kind of actions, in order to avoid idle conditions of the human.
- **Human action**: is executed by the human alone. Since the human is an external agent, and therefore considered " uncontrollable ", this action cannot be controlled: it can only be predicted by a predictive algorithm (developed in [51], which is out of the scope of this thesis) that allows to approximately know the duration of such action.
- **Human to Robot action**: is an action performed by the human only - so still uncontrollable - but followed by a robot action.

The TPN is built accordingly to the presented activity classification, using different types of places and transitions.
In particular, the places will be distinguished in:

- **Action in progress**: the time spent by a token in this place identifies the duration of the corresponding action, i.e. the action started when the transition in input to this place fired.
- **Resource**: a token in this place represents the availability of the resource associated to it. The resource can be the robot (or one of its arms, since the arms work independently and are considerable as different agents), the human or a buffer.
- **Biding place**: it is always subsequent to a " Resource place "(related to the availability of human or robot): a token in this place means that the agent (human or robot) is available but not ready to perform an action (due to lack of resources; e.g. in an assembly process, the agent that is ready misses the piece that is supposed to be machined) and, therefore, is in a waiting condition.

Concerning transitions, the following distinction is made:

- **Controllable**: is associated to the command to give to the robot, in order to let it start the intended action. A controllable transition is the outcome of a decision taken by the scheduler and so there is no delay to be considered. In other words, it fires instantaneously.

- **Uncontrollable**: is related to an action and so is characterized by a duration. The latter depends on the time required to perform the action and is represented in such a way to account for the uncertainty on its duration (representation showed in Chapter 4)

When modeling the system as a FTPN, every action performed by robots is modelled as a controllable transition followed by an uncontrollable one. This choice arose with the need of maintaining the control on the succession of actions, each one with its own uncertain duration.
Consider for example a system in which the robot can perform two actions $R_1$ and $R_2$; suppose that the system is in a condition for which both actions can be executed. Translating the situation in a FTPN representation, it means that both the uncontrollable transitions associated to $R_1$ and $R_2$ (namely, $t_{U_1}$ and $t_{U_2}$ respectively) are enabled and can fire.
Scheduling has the aim of choosing which one to fire: the choice depends on the system specifications. For instance, if the goal is to have the minimum execution time, it might be more convenient to choose $t_{U_1}$ over $t_{U_2}$. Nevertheless, since $t_{U_1}$ and $t_{U_2}$ are uncontrollable transitions, there is no power of choice: the system evolves uncontrollably and, likely, without satisfying the criteria of preference.
To avoid this problem, controllable transitions are introduced: $t_{U_1}$ is preceded by $t_{C_1}$ and $t_{U_2}$ by $t_{C_2}$. Hence, it is possible to choose to fire $t_{C_1}$ over $t_{C_2}$: in this way, the system is brought into a configuration for which it is straightforward that $t_{U_1}$ would be the next transition to fire (supposing that there are no other transitions in the system and the firing of $t_{C_1}$ disables $t_{C_2}$).
An example of controllable and uncontrollable transitions alternating is presented in Figure 3.1.

Considering a generic system of collaborating agents, it is possible to reach a condition for which more actions can be executed. In such cases, the scheduler has to choose the most appropriate action, by choosing the controllable transition that would lead to the most convenient uncontrollable future evolution.
In other words, controllable transitions allow to bring the system in a situation in which only a group of certain uncontrollable transitions can fire and, so, only a group of certain future scenarios are possible.

# Chapter 4

# Uncertainty: Possibility and Fuzzy Theory

In this Chapter all the main notions of Possibility and Fuzzy Theory are provided, together with the reasons that led to tackle the problems considered in this thesis with a fuzzy approach.
Section 4.1 describes in detail the problem concerning the duration of actions in a HRC context. In Section 4.2 a brief historical excursus is presented on the development of Fuzzy Theory and the diffusion of the Possibility Theory to treat problems in real systems. Section 4.3 introduces the problem of epistemic uncertainty and the reasons that led to the choice of this thesis to use a method to model uncertainty, that is an alternative to the traditional ones. In Section 4.4 the basic concepts of Fuzzy Theory are illustrated and in Section 4.5 concepts of Possibility Theory are presented. Section 4.6 deals with the problem of uncertainty propagation, solved with a possibilistic approach. In Section 4.7 the relations between possibilistic and probabilistic approach are illustrated.

## 4.1 Managing actions with uncertain duration

As already stated in Chapter 3, the uncertainty that characterizes the duration of the actions of agents is a relevant issue. Indeed, the scheduling algorithm that chooses the actions to be sent to the robot has to take into account the uncertainty on the duration of each of such actions.
In fact, if the decision making is performed neglecting the uncertainty that affects the system, an apparently optimal plan might become unfeasible or at least considerably different from the real evolution of the system.

Moreover, the expected times at which the operations are actually performed would turn out to be far from the real ones and so the overall performance would then be distant from the forecast one, leading to an ineffective scheduling.
According to what has been stated above, the work here presented is aimed at properly choosing the best plan, interpreting the uncertainty as a powerful source of “ information ”. Hence, the purpose is to compute a reachability tree in which the uncertainty is embedded in the firing times of the transitions. This would reflect the uncertainty on the duration of actions and would allow the propagation of such uncertainty from one action to the following one. By doing so, the overall uncertainty affecting the whole system can be properly modelled.
In the classes of Petri nets considered by this work, every transition is linked to a physical action whose uncertain duration must be modelled. To this aim, the actions assigned to the agents are managed in different ways:

- Human action: the external algorithm - presented in [51] is adopted for predicting the start time of such actions, i.e. provides a prediction of the human intention.
- Robot action: the duration of the actions of the robot is computed basing on data collected from past executions of the algorithm. The collected data is used as a prediction of the future duration.

The scope of the thesis is to model the “ measurement imprecision ” - in terms of “ how much is not known about the measure ” - to guarantee the efficiency of the scheduler of robotic actions.
Once the uncertainty on every single action is modelled, it is important to study how it is propagated through all the sequential actions, i.e. throughout the reachability tree.
Consider as an example a system, where an agent (e.g. a robot) has to perform 3 actions on a working piece:

1. take a piece from the buffer
2. work the piece
3. place the piece on a pallet

Each of these actions has a duration that is approximately known because it is impossible to have a precise knowledge of this quantity. To predict when the robot would complete the whole action, i.e. when the piece would be on the pallet, the total uncertainty must be considered. Such total uncertainty is supposed to consider and combine all the single uncertainties, assuming in addition that each of them propagates, influencing the successive ones. In the following, a computation for the example is reported to show two different scenarios and in particular what happens when the uncertainty is not propagated through sequential actions:

- First scenario: the first two actions are “ perfectly known ” - and so their duration is a sharp value - whilst only the last one is not completely known (duration defined over a temporal interval). Suppose that “ place the piece on a pallet ” requires between 4 and 5 s; “ take a piece from the buffer ” takes 5 s and “ work the piece ” 90 s. Consequentially, the total action is completed within 99-100 s, i.e. result of (90+5+“ number comprehended between 4 and 5 ”).

- Second scenario: the duration of every action accounts for imprecise information and so is defined over a temporal interval. Assume that:

    1. “ take a piece from the buffer ” takes from 5 s to 8 s
    2. “ work the piece ” from 85 s to 95 s
    3. “ place the piece on a pallet ” 4 s to 5 s. Then, the time required to perform the whole action would be encapsulated in a range, with lower bound computed as “ best case ” (5+85+4 = 94 s) and upper bound as “ worst case ” (8+95+5 = 108 s). The total action has a duration comprehended in the interval [94, 108].

Notice that the magnitudes of the ranges obtained in the two cases are rather different: where propagation of the uncertainty is considered, the resulting interval is much wider.
As a consequence, not considering such uncertainty would lead to a very approximated and reductive result. Therefore, it is essential to account for the imprecision, as it occupies a leading role in the evaluation of an optimal plan.

In the work developed in this thesis, Fuzzy Theory will be used to model and propagate the uncertainty. Thus, in the following a brief background and notions of such theory will be given.

## 4.2 From “ Theory of Communication ” to “ Fuzzy Information Theory ”

In [58], Shannon published a milestone regarding information theory: the intention was to define a general theory on communication, able to account for the noise on transmission channels and to preserve the nature of the original transmitted message as much as possible.
The communication model considered by Shannon [61] consisted of:

- Source of information that produces the messages to be sent to a receiver, through a communication channel
- Transmitter to transform the messages, in order to produce signals to be sent across the channel (e.g. in telephony, acoustic pressure is converted into an electric current)
- A channel as means for transmission (e.g. wires, light beams, etc.)
- A receiver that performs the opposite operation of the transmitter: reconstructs the original message from the signal received
- Destination, i.e. person or entity entitled to receive the message

According to this model, Shannon aimed at transmitting a message the least possible affected by errors with respect to the original message: he claimed that the cause of concern in communication was to reliably reproduce “ either exactly or approximately ” at one point a message selected at another point [58].

However, he was not concerned about controlling that the transmitted message was meaningful. Indeed, as later on stated the mathematician and physicist Warren Weaver, even a message without transmission errors could lack of real information, if the content is non-sense: when two messages are received, they can be interpreted as equivalent even though one is “ heavily loaded with meaning and the other is pure nonsense ” [59].

In July 1949, Weaver published “ The Mathematics of Communication ”: in the introduction, he explained the reasons for Shannon’s mathematical theory was not enough to describe the imprecision of the information transmitted. According to him, such theory considered the “ correctness ” only in technical terms, whilst also “ semantic ” and “ influential ” problems of communication should have been a matter of concern.

He believed that - when some noise is introduced - the received message contains some distortions and errors, i.e. unknown and exogenous material. Then, to remove the spurious part of the message it is first needed to determine the amount of “ ambiguity ” brought by the noise itself. Such quantity is called by Weaver “ equivocation ”. [60]
Weaver soon realized that communication had to be improved trying to solve not only technical problems, related to accuracy of transmission from one transmitter to a receiver, but also those related to the interpretation of the meaning attributed to the received message.
He defined the problems concerned with interpretation [61], dividing them into:

- Semantic problems: about how the receiver interprets the meaning of the received message; then such interpretation has to be compared with the intended meaning of the sender
- Influence (or effectiveness) problems, concerned with the success with which the meaning conveyed to the receiver leads to the desired conduct on his part

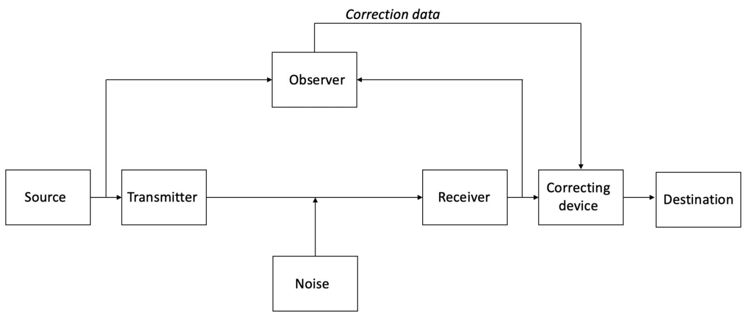


**Figure 4.1:** Shannon’s communication model, [61]

According to Weaver, the Theory of Communication could be enlarged adding to Shannon’s model (Figure 4.1) two more elements, that would have accounted for semantic problems (Figure 4.2).
The first of these blocks, in Figure 4.2 illustrated as “ Semantic Noise ”, was inserted between the source of information and the transmitter. This block had to codify the distortions introduced by the source, that are different from the noise introduced by the transmission channel.
The errors introduced in this phase of communication and transmitted together with the message were then eliminated by the specular block “ Semantic Receiver ”, inserted between the receiver and the destination.

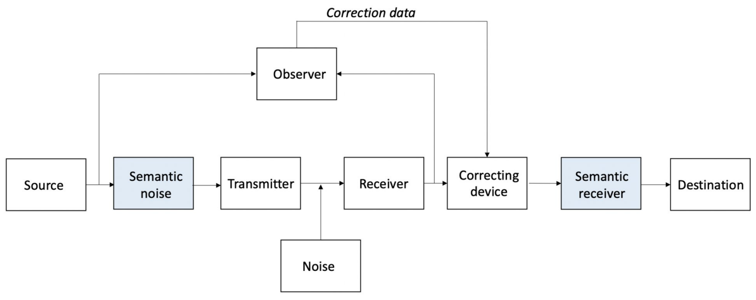


**Figure 4.2:** Weaver's communication model: an extension of Shannon's model, [61]

This block actuated a procedure of decoding, to get rid of the semantic errors. The process of recovery was executed basing on the public of receivers, to which the message was addressed.

This established the basic concepts for the lately developed theory, founded on the ambiguity of language (vagueness) and on the importance of the meaning of the transmitted information, i.e. the Fuzzy Theory.

In [63], Zadeh started dealing with recovery process of transmitted signals. At first, influenced by Shannon's theory, he studied a technique to accurately reconstruct the transmitted messages, introducing the concept of " distance ". Given a finite set of possible signals to send to the receiver:

$$X = \{\, x_k(t) \,\} \qquad k = 1, 2, \ldots, n$$

calling $y$ the signal that reaches the receiver, $y$ would be the signal used to reconstruct the original message. To do so, the receiver compares $y$ with all the possible $x_k$, using its new concept of distance function, $d(x, y)$. Eventually, the receiver selects the signal $x_k$ that is closer to $y$, i.e. the one with the smallest distance function. Overall, the received signal is assumed to be the $x_k$ for which it holds that:

$$d(x_k, y) < d(x_i, y) \qquad i \neq k, \qquad \forall k, i$$

After a while, Zadeh himself realized that it could often turn out to be inconvenient or even impossible to determine a quantitative measure, to compare two signals.
What mostly characterizes the ambiguous predicates is that their structure and the information that they carry is extremely complex and so not easily reducible to purely " extensional logic " notions. This idea provides the

foundations for the lately developed “ Theory of Fuzzy sets ”, [63], that was introduced as a consequence of the need to represent a vague predicate: it enriches the purely extensional concepts with membership degrees.
After developing such extension of classical sets, also arithmetic, logic and measures were extended in this direction, by implementing techniques able to represent concepts in a way analogous to how humans would represent such concepts.

The problem of interpretation of the meaning, posed by Weaver, could be seen as a fuzzy process too: the encoding executed by the “ Semantic Noise ” stands for a fuzzification process (i.e. process that translates the available information in a representation - namely, fuzzy set - that accounts for the uncertainty affecting such information), whereas the block “ Semantic Receiver ” could be seen as a defuzzification process. The fuzzy set obtained before was now converted into an output variable, that therefore resulted in the same form of the input one. Such process was aimed at obtaining an output information the most possible similar to the one that the sender aimed at transmitting.

Nowadays, fuzzy models are used for control applications and for all those fields in which it is required to define robust models starting from qualitative information or imprecise and ill-known data.
As it will be later shown, this thesis considers the information available as imprecise. Thus, such information is “ fuzzified ”, i.e. represented in terms of fuzzy sets. After few computations, the algorithm outputs are still fuzzified; therefore, it is required to extract the information to be sent to the intended receiver in a proper way. Depending on what the receiver is (in this case the final user of the extracted information is the robot), the data have to be “ defuzzified ” accordingly. In fact, the information eventually obtained is suitable to be sent as a command to the robot.
In the following Section, a general insight on the problem of the uncertainty is given, presenting what it consists of and the available approaches to counteract it.

## 4.3 Modeling uncertainty

Considering a generic system (Figure 4.3) with input parameters affected by uncertainty, it is important to know and propagate such uncertainty in order to obtain an output “known” as much as possible, i.e. that can be described as much precisely as possible.

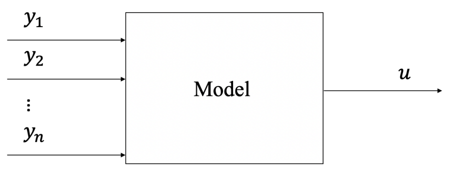


**Figure 4.3:** System with uncertain input parameters

First of all, it is necessary to understand the causes of the uncertainty. Depending on the factors that brought to an imprecisely known system, it is possible to classify the uncertainty in two main kinds:

- **Aleatory uncertainty**: it is due to natural variability of the physical world, so it is a direct consequence of the inherent randomness in nature. It exists regardless of human's knowledge.
  For example, when flipping a coin, it comes up heads or tails randomly. Even after performing many experiments and having computed the probability of coming up heads, it is still not possible to predict the exact outcome of the next experiment.
  This kind of uncertainty cannot be eliminated nor reduced by retrieving more knowledge or information. For this reason, it is treated with Probability Theory. Aleatory uncertainty is often also referred to as " randomness ".

- **Epistemic uncertainty**: it origins from human's lack of knowledge of the physical world and from the inability of precisely measuring and modeling the real world. Unlike aleatory uncertainty, by acquiring more knowledge on the problem the epistemic uncertainty can be considerably reduced. For example, the distance between Milan and Rome can be estimated more precisely if the distance from Milan to Bologna is known. Epistemic uncertainty is also called " fuzziness " [62].

The quantities involved in the HRC scheduler developed in this thesis are assumed to be part of the category of epistemic uncertainty. For this reason, the following Sections will show how to model and propagate uncertainty, assuming it to be epistemic.

### 4.3.1 Epistemic uncertainty

As briefly shown in Figure 4.4, there are two main techniques to treat epistemic uncertainty: if there are sufficient statistical data, their

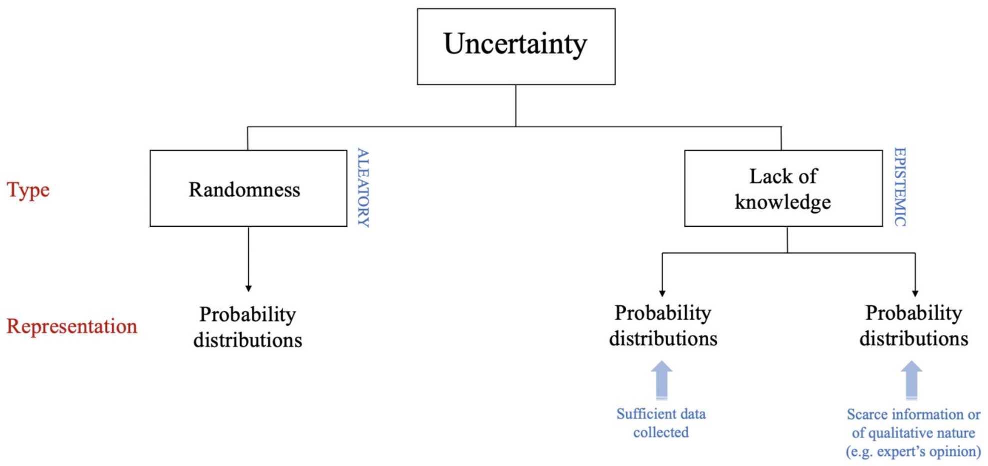

Uncertainty
Type
Randomness
ALEATORY
Lack of knowledge
EPISTEMIC
Representation
Probability distributions
Probability distributions
Probability distributions
Sufficient data collected
Scarce information or of qualitative nature (e.g. expert's opinion)

**Figure 4.4:** Types of uncertainty and possible representations

uncertainty can be represented using Probability Theory. If the information available is scarce and/or is of qualitative nature, rather than quantitative, an alternative approach is recommended. As it will be shown later, an approach that leads to satisfactory results makes use of Possibility Theory. An overlook on the two methods is given in the following:

1. Sufficient informative data: use probabilistic approach
   Consider the generic SISO system of Figure 4.5, where $y$ is a random variable described by the probability distribution $f(y)$. To build $f(y)$, it is necessary to observe $y$ performing many experiments and so collecting enough data.
   Suppose to repeat the experiment $N$ times: the information collected is $(y_1, y_2, \ldots, y_N)$. From this, the probability distribution function

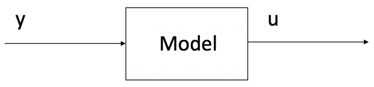


**Figure 4.5:** Generic SISO system

and the cumulative distribution function are built (Figure 4.6). These probability distributions give a representation of the uncertainty related to the random variable $y$. For instance, the uncertainty on "does $y$ belong to $[y_1, y_2]$?" is expressed by the value of the area under the probability distribution function, between those boundaries (shaded area in Figure 4.6):

$$P\{\, y \in [\, y_1, y_2 \,] \,\} = \int_{y_1}^{y_2} f(y)\, dy \tag{4.1}$$

The uncertainty about “ does $Y$ precede $y_2$? ” is expressed by (4.2),

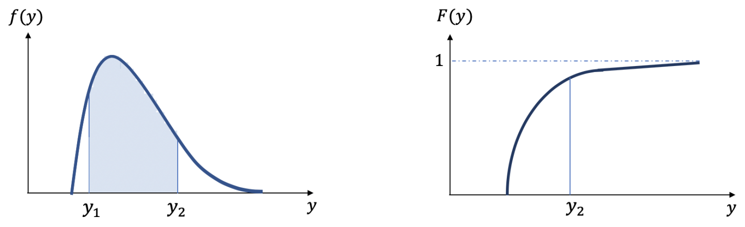


**Figure 4.6:** Probability distribution and cumulative distribution of a generic random variable $y$ [64]

that turns out to be an integral over an open shape, hence not always easy to solve in an analytical way.

$$F(y_2) = P(\, Y < y_2 \,) = \int_{-\infty}^{y_2} f(y)\, dy \tag{4.2}$$

2. Scarce information of the parameters and/or information of qualitative nature: use possibilistic approach
To prove that in such a scenario it is not convenient to use a probabilistic approach, two examples are provided.

**Example 1:** consider a limit case in which no information is given about a parameter $y$, except that its value is located somewhere between $y_{min}$ and $y_{max}$.
With a probabilistic approach, the lack of knowledge leads to “ every value is plausible ” and therefore equiprobable: the probability of every value comprehended in $[y_{min}, y_{max}]$ is $\frac{1}{y_{max}-y_{min}}$ (Figure 4.7).
As a consequence, $y$ would be taken as the mean value of all these probable values:

$$y_m = \frac{y_{max} - y_{min}}{2}$$

This implies:

$$P(\, y \in [\, y_{min}, y_m \,] \,) = P(\, y \in [\, y_m, y_{max} \,] \,)$$

which is a paradox, since there is no information available and hence no relation between $P(y \in [y_{min}, y_m])$ and $P(y \in [y_m, y_{max}])$ should

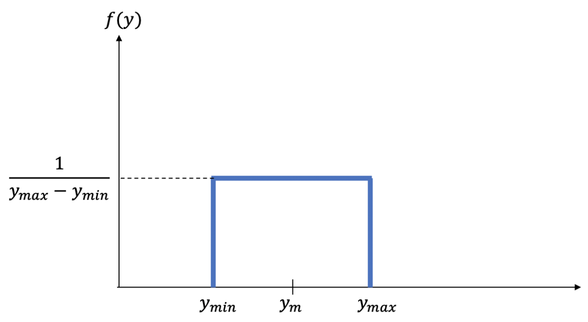


**Figure 4.7:** Probabilistic approach: not available information approximated to uniform distribution [64]

be deduced.

**Example 2:** consider a source that provides non-quantitative information, e.g. an expert that says " the value $y$ is between 9 and 11, with a preference for 10 ". Trying to catch such information in terms of probability distribution is hard (Figure 4.8).

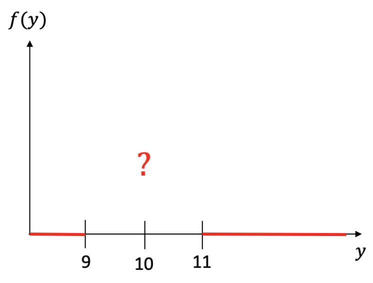


**Figure 4.8:** No way to represent *qualitative information* as a probability distribution [64]

Here appears the necessity to develop a theory that allows to represent this type of information. Zadeh introduced in [63] a solution to this problem, by means of Possibility Theory, that was based on the meaning of the information rather than on the measure of the information. He himself declared that whenever the aim is to tackle the meaning of information, rather than its measure, then the analysis is " possibilistic rather than probabilistic in nature ".

The mathematical approach that forces the human lucubration to a binary reasoning, i.e. what brought to the leading theories up to then (e.g. Shannon's theory, Section 4.2), doesn't reflect the human's capability of thinking articulately. The Possibility Theory instead gives the means to “ translate ” - in mathematical terms - the mental elaborations of human beings, catching the fuzziness, interpreted as concept of “ blurred information ” or “ vagueness ”, intrinsic in such elaborations.

## 4.4 Fuzzy Theory

In the following paragraphs, basic concepts of Fuzzy Theory will be illustrated.
The concepts provided in this Section explain the logic and operations adopted by this thesis: in particular, they will be necessary to properly show the reasoning developed in the thesis and later explained in Chapter 5.

A **fuzzy set** is a set whose elements are characterized by a membership degree, that takes any value between 0 and 1. A fuzzy set $A$ is therefore defined by a membership function:

$$\mu_A : X \to [\,0,1\,]$$

where $X$ is the universe in which $A$ is defined. The universe $X$ is a conventional set, alternatively said “ crisp ” set: every element either belongs to $X$ or does not, i.e. every element has a crisp membership degree that is 0 or 1.

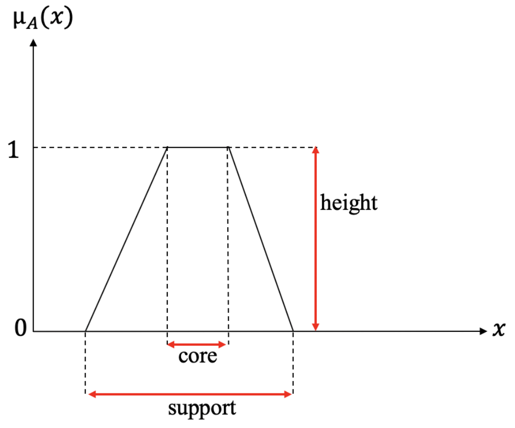


**Figure 4.9:** Height, core and support of a fuzzy set [64]

The *height* of a fuzzy set $A$, $hgt\,(A)$, is defined as:

$$hgt\,(A) = \sup_{x \in X} \mu_A(x)$$

and fuzzy sets with a height equal to 1 are said “ normal ”, whilst sets with height less than 1 are called “ subnormal ”.
The *core* of a fuzzy set, also referred to as *kernel*, is a crisp subset of $X$:

$$core(A) = \{\, x \in X \,|\, \mu_A(x) = 1 \,\}$$

The *support* of a fuzzy set is a crisp subset of $X$:

$$supp(A) = \{\, x \in X \,|\, \mu_A(x) > 0 \,\}$$

These parameters are represented in Figure 4.9.
A *fuzzy singleton* is a fuzzy set whose support is a single point. It might be useful to gather in a unique set all the values characterized by a membership degree greater or equal to a specific threshold. Such idea is expressed by a particular set, called *alpha-cut*:

$$A_\alpha = I = 1\{\, y : \pi(y) \geq \alpha \,\} \tag{4.3}$$

in which $\alpha \in [0, 1]$ represents the desired threshold.
For instance, $A_{0,30} = [\, y_1, y_2 \,]$ is the interval containing all the elements $y$

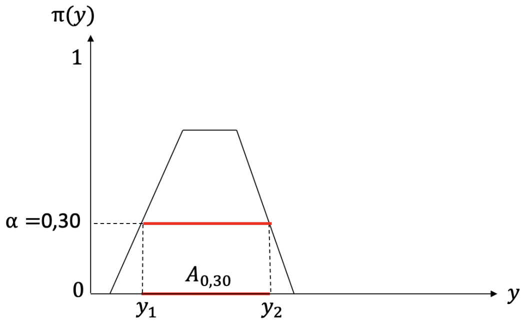


**Figure 4.10:** Alpha-cut containing all the elements with membership degree at least equal to 0.30 [64]

with membership degree at least equal to 0.30 (Figure 4.10).
A fuzzy set defined in $\mathbb{R}$ is said to be *convex* if all its $\alpha - cuts$, with $\alpha \in [\, 0, 1 \,]$ are canonical convex sets (Figure 4.11).

### Description of a fuzzy set

Be $A$ a subset of a universe of discourse $X$:

- Discrete universe $X = x_1, x_2, \ldots$

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \ldots \right\} = \left\{ \sum_i \frac{\mu_A(x_i)}{x_i} \right\} \tag{4.4}$$

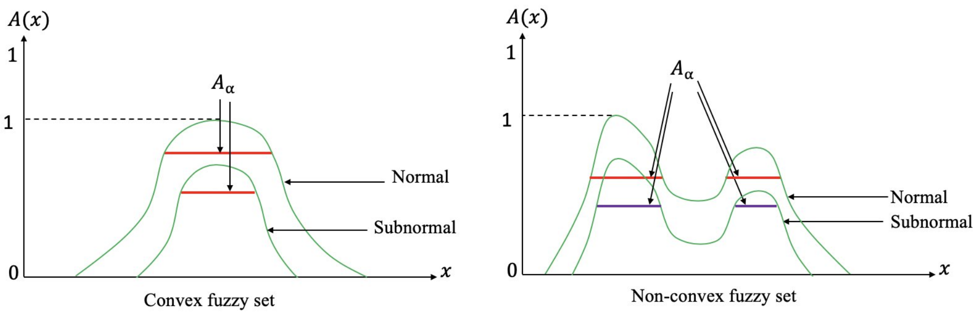


**Figure 4.11:** Convex and non-convex fuzzy sets [75]

Eq. (4.4) shows that a generic $x_i$ belongs to $A$ with membership degree $\mu_A(x_1)$; note that the symbol "/" doesn't represent a division but a comparison term, with the meaning of "at point $x_i$ corresponds a membership degree $\mu_A(x_i)$ ".

- Continuous universe $X$

$$A = \left\{ \int \frac{\mu_A(x)}{x} \right\} \tag{4.5}$$

**Operations on fuzzy sets**

Be $A$ and $B$ two arbitrary fuzzy subsets of a universe $U$, then, the following operations can be performed [75]:

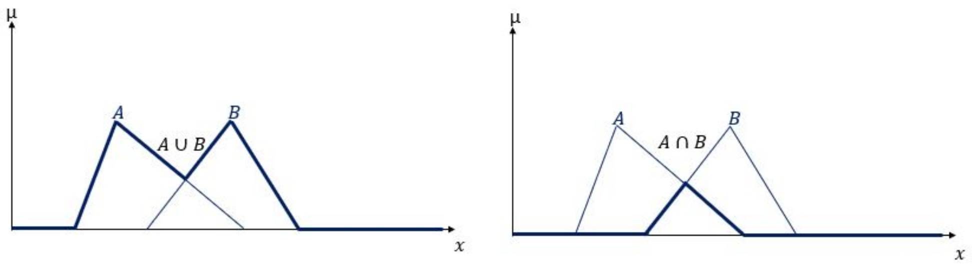


**Figure 4.12:** Union and intersection of fuzzy sets [75]

- Union of fuzzy sets (Figure 4.12):

$$\mu_{A\cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right) \tag{4.6}$$

- Intersection of fuzzy sets (Figure 4.12):

$$\mu_{A\cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right) \tag{4.7}$$

- Complementary of a fuzzy set (Figure 4.13)

$$\bar{A}(x) = 1 - A(x) \tag{4.8}$$

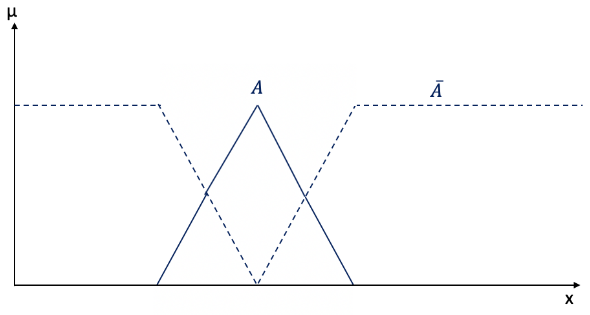


**Figure 4.13:** Fuzzy set and its complement [75]

Whilst the operations on canonical sets, including De Morgan laws, hold for fuzzy sets, the laws of " tertium non datur " and " non contradiction principle " are not valid in Fuzzy Theory (4.4). In fact, be $A$ a fuzzy subset of the universe $X$ and be $\bar{A}$ the complement of $A$. When joining $A$ and its complement the set obtained is not the universe $X$. Similarly, when intersecting $A$ and its complement, the result is not the empty set (Figure 4.14):

$$A \cup \bar{A} \neq X$$
$$A \cap \bar{A} \neq \emptyset$$

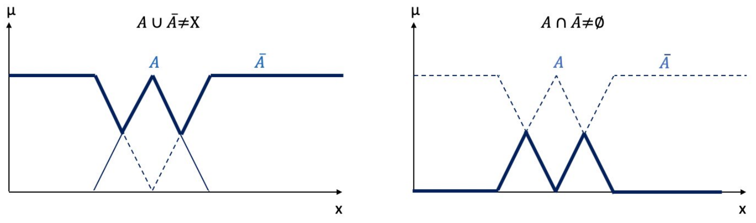


**Figure 4.14:** Union and intersection between a fuzzy set and its complement [75]

**Fuzzy relations**

The concept of " relation " plays a key role in mathematics and also for fuzzy logic. In the following, the main fuzzy relations are shown.

*Cartesian product of fuzzy sets*
Be $A$ and $B$ two fuzzy subsets of the non-fuzzy universes $X$ and $Y$ respectively. The Cartesian product $A \times B$ is a fuzzy subset of $X \times Y$, whose elements are characterized by a membership degree defined as:

$$\mu_{X \times Y} = min\,(\,A(x), B(y)\,) \tag{4.9}$$

The membership degree $\mu_{X \times Y}(x, y)$ can be interpreted as an esteem of the value that expresses the strength of the relationship between $X$ and $Y$. Such relationship is expressed by a fuzzy relation $R$, that expresses the link between the two non-fuzzy universes $X$ and $Y$.
The relation $R$ is computed as a Cartesian product, turning out to be a fuzzy subset of $X \times Y$:

$$R = \{\,\mu_R(x, y)/\ (y, x)\,\}, \forall (x, y) \in X \times Y$$

*Composition of fuzzy relations*
Be $R$ a fuzzy relation on $X \times Y$, $S$ a fuzzy relation on $Y \times Z$, and $W = R \circ S$ a " composed " fuzzy relation on $X \times Z$, then the membership degree of $\mu_W(x, z)$ is computed as:

$$\mu_W(X, Z) = \max_{y \in Y} \quad \min\,(\,\mu_R(x, y), \mu_s(y, z)\,) \tag{4.10}$$

## 4.5 Possibility Theory

To explain what a possibility distribution is, it is convenient to first introduce the concept of " fuzzy restriction ", with which the possibility distribution is strictly related.
Consider a universe of discourse $U$ and its generic element $u$. Taking a variable $X$ with values in U, the assignment:

$$X = u$$

implies that $X$ takes the value $u$, $u \in U$.
Be $F$ a fuzzy subset of $U$ (Section 4.4), it is possible to assign to every element $u \in U$ a degree that expresses the compatibility of $u$ with respect to the relation $F$, i.e. degree of truth that $X$ belongs to $F$. Such degree goes by the name of " membership degree " and is represented as a function $\mu_F(u)$.

Stating that $F$ acts as an *elastic constraint* on the values $u$ that can be assigned to $X$, i.e.:

$$X = u : \mu_f(u) \tag{4.11}$$

means that the assignment " $X = u$ " is characterized by a degree of truth (or membership). This implies that F restricts the set of values that $X$ can assume.
A clarifying example is given in the following.

**Example**: take as universe of discourse the set $\mathbb{N}$ of natural numbers and consider a fuzzy subset of $\mathbb{N}$. $F$ is a set of small numbers, where " small " is a qualitative parameter, and can be defined as a fuzzy parameter.
Every element $u \in \mathbb{N}$ can potentially belong to $F$, since the definition of " small " is ambiguous: each of them can be considered small with a certain degree of truth.
Suppose, for instance, that $F$ imposes the following constraints:

1. $u$ is small $\implies$ $u \in \{0, 1, 2, 3, 4\}$
2. $u = 0$ and $u = 1$ are the points with maximum membership degree
3. $u \in [\,1, 4\,]$ have a decreasing membership degree

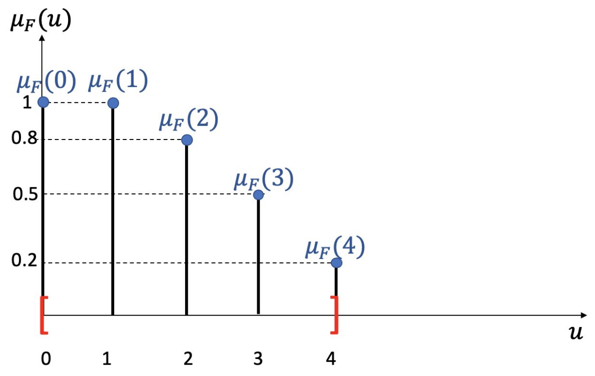


**Figure 4.15:** Determine membership degree of every element of the universe of discourse

Assigning numerical values: $\mu_F(0) = 1$, $\mu_F(1) = 1$, $\mu_F(2) = 0.8$, $\mu_F(3) = 0.5$, $\mu_F(4) = 0.2$, $\mu_F(u) = 0\ \forall u > 4$. Such values are represented in Figure 4.15 and then interpolated to obtain the possibility distribution of Figure 4.16.
Since $\mu_F$ is a fuzzy distribution, is also a closed set (see Section 4.4),

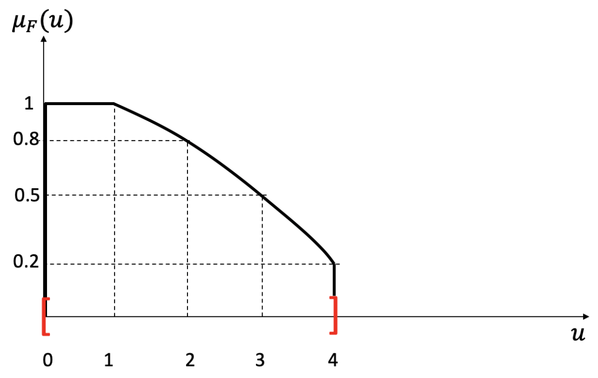


**Figure 4.16:** Create the fuzzy distribution

limiting the values that $X$ can effectively assume in $U$. Therefore, $F$ is said to be a *restriction* on $X$.
Moreover, Eq. (4.11) implies that to be able to assign $X = u$, i.e. to guarantee that exists a minimum compatibility of $u$ with the relation $F$ - or, in other words, to guarantee that it is at least minimally true that $u$ belongs to $F$ - $u$ must have a membership degree at least equal or greater than $1 - \max(\,\mu_F(u)\,)$. Membership degrees lower than such value indicate that there is no compatibility between $u$ and $F$. The above-mentioned expresses the concept of " necessity ".

Be $R(x)$ the *fuzzy restriction* associated to $X$, the relation:

$$R(x) = F \tag{4.12}$$

shows that $F$ is a fuzzy restriction on $X$, and therefore represents the assignment of a fuzzy set - meant as distribution of a membership degree $\mu_F$ - to the restriction associated to $X$, i.e. $R(X)$.

In the following, the previously presented example is resumed, to better explain the concept that links a fuzzy restriction to a possibility distribution.
Consider the set of natural numbers $\mathbb{N}$ and a fuzzy subset of $\mathbb{N}$, $F = \{$ set of small numbers $\}$. Then, every element $u \in \mathbb{N}$ satisfies the condition "$X$ is $F$" - namely, " $X = u$ is small " - with a certain degree of truth, $\mu_F(u)$.
Here the point is to investigate how to compute the possibility that $X$ assumes a particular value $u$, for instance $X = 2$.
Since for the reported example $\mu_F(2) = 0.8$, i.e. the degree of compatibility

between 2 and the concept of “ small ” is 0.8.
The statement “ $X$ is small ” converts the meaning of 0.8 from *degree of compatibility* between “ 2 ” and “ *small* ”, to *degree of possibility* that $X$ is effectively 2, since “ 2 is small ”.
In short, the compatibility of a value $u$ with *small* is converted into possibility that $X$ is equal to $u$, since “ $u$ is small ”.
It follows that the *possibility distribution function* associated with $X$ is denoted by $\pi_X$ and is defined to be numerically equal to the membership function of $F$:

$$\pi_X \triangleq \mu_f \tag{4.13}$$

Thus, $\pi_X(u)$ is the possibility that $X = u$ and is postulated to be equal to $\mu_F(u)$.
Hence, assigning a possibility degree to the variable $X$, corresponds to restrict $X$ itself to the only values that are compatible with the restriction:

$$\Pi_x = R(x) \tag{4.14}$$

$X$ is therefore associated with a possibility distribution $\Pi_x$, and according to Eq. (4.12), it is possible to state that:

$$\Pi_x = F \tag{4.15}$$

i.e. the possibility distribution is equal to the fuzzy set of interest.

**Important remarks:**

1. Eq. (4.14) implies that the possibility distribution $\Pi_X$ can be considered as an interpretation of the concept of fuzzy restriction. Therefore, the theory of fuzzy sets provides the basis for a mathematical treatment of possibility distributions.

2. The association between possibility distribution and fuzzy restriction is an intuitive consequence of the above-treated reasoning. Such association allows to treat complex concepts of Possibility Theory with much easier operations, taken from Fuzzy Theory. For instance, the concepts of *conjunction* and *disjunction* of propositions in the form “ $X$ is $F$ ”, that are difficult to treat because of the language ambiguity, can be “ translated ” in basic operations on fuzzy sets, like *union* and *intersection* respectively (Section 4.4).

3. The definition of $\pi_X(u)$, in Eq. (4.13), highlights that the degree of possibility can be any of the values in $[\,0, 1\,]$, rather than only 0 or 1.

Indeed, fuzzy sets fully express the concept of " blurred ", in contrast with the one of " crisp " represented by the canonical sets.
In the latter, considering a set $A$ defined in a universe $U$, where $U$ has elements $x$, it is possible to distinguish whether $x$ belongs to $A$ or doesn't. Thus, the membership degree, $\mu_A(x)$, of $x$ related to $A$ is a Boolean number:

$$\mu_A(x) = \begin{cases} 1 & \text{iff } x \in A \\ 0 & \text{iff } x \notin A \end{cases} \tag{4.16}$$

If $A$ is instead a fuzzy set, it wouldn't be appropriate to state that $x$ either belongs or does not belong to $A$. Indeed, in applications where it is not clear if $x$ belongs or not to the above-mentioned set, it is more correct to state that $x$ belongs to $A$ with a certain degree of truth (membership degree).

### 4.5.1 Possibility measures

A further distinction between probability and possibility can be given through a parallelism between the concept of possibility measure and its analogous regarding probability.

- Consider $A$ to be a non-fuzzy subset of a universe of discourse $U$ and be $\Pi_X$ the possibility distribution associated to the variable $X$ defined in $U$. Then, the possibility measure of $A$, $\pi(A)$, stands for the possibility that $X$ belongs to $A$, and it's a real number defined in $[0, 1]$, expressed in the following form:

$$\begin{aligned} Poss\{\, X \in A \,\} &\triangleq \pi(A) \\ &\triangleq \sup_{u \in A} \quad \pi_x(u) \end{aligned} \tag{4.17}$$

  where $\pi_X(u)$ is the possibility distribution function of $\Pi_X$.

- Consider $A$ as fuzzy subset of $U$ e and be $\Pi_X$ the possibility distribution associated to a variable $X$ with values in $U$. In this case, it is not possible to firmly state that $X$ belongs or does not belong to $A$: such proposition is rather characterized by a degree of truth.
  Thus, the possibility that $X$ belongs to $A$, i.e. the possibility measure

of $A$, is:

$$\begin{aligned} Poss\{\, X \text{ is } A \,\} &\triangleq \pi(A) \\ &\triangleq \sup_{u \in U} \quad \mu_A(u) \wedge \pi_x(u) \end{aligned} \tag{4.18}$$

where “ $X$ is $A$ ” replaces “ $X \in A$ ” that was used in Eq. (4.17); $\mu_A$ is the membership function of $A$ and $\wedge$ is the symbol for *min*.

According to the definitions given, the possibility measure of union or intersection of two fuzzy sets can be given.

Be $A$ and $B$ two arbitrary fuzzy subsets of U, then the possibility of the union of two sets is:

$$\pi(A \cup B) = \pi(A) \vee \pi(B) \tag{4.19}$$

The probability measure of the union would instead be:

$$P(A \cup B) \leq P(A) + P(B) \tag{4.20}$$

provided that $A \cup B$ exists.
In addition, if $A$ and $B$ are disjoint (i.e. $\mu_A(u) \cdot \mu_B(u) \equiv 0$), it holds:

$$P(A \cup B) = P(A) + P(B) \tag{4.21}$$

Eq. (4.21) represents the “ additivity property ” of probability measure. On the contrary, the additivity property doesn’t hold for possibility measures. Nevertheless, the property of possibility measure in (4.19) can be seen as the equivalent of (4.21), where the sign “ $+$ ” is replaced by the *max* operator( “ $\vee$ ”).
Similarly, the possibility measure of the intersection between $A$ and $B$ reflects the following property:

$$\pi(A \cap B) \leq \pi(A) \wedge \pi(B) \tag{4.22}$$

Whenever $A$ and $B$ are non-interactive (i.e. $\mu_A(u) \cdot \mu_B(u) \equiv 0$) - in other words, when their possibilities don’t affect each other - the inequality becomes an equality [67]:

$$\pi(A \cap B) = \pi(A) \wedge \pi(B) \tag{4.23}$$

On the other side, the probability measure of such intersection would be:

$$P(A \cap B) \leq P(A) \wedge P(B) \tag{4.24}$$

and, if $A$ and $B$ are independent and non-fuzzy:

$$P(A \cap B) = P(A) \cdot P(B) \tag{4.25}$$

As for the union, it is noticeable that (4.23) is equivalent to (4.25), where the multiplication sign is replaced by the *min* operator( “ $\wedge$ ”).

### 4.5.2 Fuzzy numbers

A fuzzy number is a specific kind of fuzzy set. It is possible to state that a fuzzy set $F$ is also a fuzzy number if the following conditions are met [75]:

- $F$ is convex
- $F$ is a normal fuzzy set, i.e. $hgt\,(F) = 1$
- the membership function of the fuzzy set is piece-wise continuous

Thus, a fuzzy number must have an increasing curve that goes from a value of possibility equal to 0, up to possibility 1, and then a decreasing part that ends in a value with possibility 0.
Examples of fuzzy numbers are triangles and trapezoids.
In Figure 4.17, on the left an example of fuzzy number, which is normally denoted as $A = (a_1, a_M, a_2)$; on the right, a distribution not satisfying the criteria of fuzzy numbers.

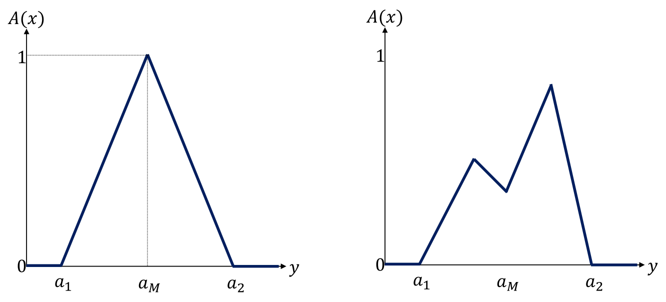


**Figure 4.17:** Triangular fuzzy number

The membership function for a triangular fuzzy set takes the analytic form of (4.26).

$$A(x) = \begin{cases} \dfrac{x - a_1}{a_M - a_1} & \text{for } a_1 \leq x \leq a_M \\ \dfrac{x - a_2}{a_M - a_2} & \text{for } a_M \leq x \leq a_2 \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

Where $a_1$ and $a_2$ identify the least possible values and $a_M$ the most likely value.

Another popular notation to identify a fuzzy number is: $A = (\underline{a}, \overline{a})$, where $\underline{a}$ is the lower limit of the number and $\overline{a}$ the upper one. In the particular case of triangles, $\underline{a}$ identifies the line connecting $(a_1, 0)$ and $(a_M, 1)$; $\overline{a}$ represents the line connecting $(a_M, 1)$ and $(a_2, 0)$.

In the following, it will be shown which basic operations it is possible to perform on fuzzy numbers. Prior to introduce an algebra for fuzzy numbers, it is needed to "extend" operations for functions defined in $\mathbb{R}$ to functions defined in the space of membership functions. Such procedure is executed by means of the extension principle [74], that gives the framework to calculate the membership degree of elements of a fuzzy set and functions of fuzzy sets, which are the result of operations.
Indeed, the *extension principle*, introduced in [68], is one of the main concepts of Fuzzy Theory. As Dubois and Prade affirmed in [65], "it provides a general method for extending non-fuzzy mathematical concepts in order to deal with fuzzy quantities".

The extension procedure consists of the following steps [76]:

- let $A$ and $B$ be two fuzzy sets, defined in universes of discourse $X$ and $Y$
- let $f$ be a non-fuzzy transformation function between universes $X$ and $Y$, such that $f : X \to Y$
- then the crisp function $f$ is fuzzified when it is intended to act on fuzzy sets defined on $X$ and $Y$.
  The fuzzy function $f : X \to Y$ indicates how the image of a fuzzy subset A of X should be computed when the function f is applied. It is expected that this image will be a fuzzy subset of Y.

From the definition given in (4.4), $A$ can be expressed as:

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots \frac{\mu_A(x_n)}{x_n}$$

The extension principle states that the image of $A$ under the mapping $f(\cdot)$ can be expressed as another fuzzy set $B$:

$$B = f(A) = \frac{\mu_A(x_1)}{y_1} + \frac{\mu_A(x_2)}{y_2} + \dots \frac{\mu_A(x_n)}{y_n} \tag{4.27}$$

where $y_i = f(x_i)$.
In general, if $f(\cdot)$ is a *many-to-one* mapping, so there exist $x_1, x_2 \in X$,

$x_1 \neq x_2$, such that $f(x_1) = f(x_2) = y^*, y^* \in Y$, then the membership degree at $y = y^*$ is the maximum of the membership degrees at $x_1$ and $x_2$. The procedure is shown in Figure 4.18 and the membership function of the extended set $B$ is:

$$\mu_B(y) = \max_{x=f^{-1}(y)} \mu_A(x) \tag{4.28}$$

Examining at the axes on the top right of Figure 4.18, it can be noted

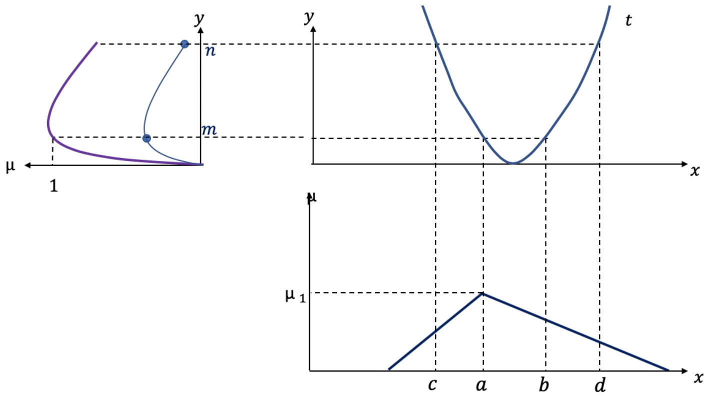


**Figure 4.18:** Extension principle: procedure [64]

that $a$ and $b$ lead to the same value $m$ on $Y$ ($f(a) = m$ and $f(b) = m$). Similarly, $c$ and $d$ lead to the same output value $n$ ($f(c) = n$ and $f(d) = n$). To create the possibility distribution of the output function, the membership degree of $m$ and $n$ must be assigned. Then, the procedure has to be repeated for every $x$.

Each of the values $m$ and $n$ will have its own membership degree, that would coincide with the maximum within the input values from which $m$ and $n$ derive ($a$ or $b$ for $m$, $c$ or $d$ for $n$).

A formalization of this concept is now provided.

The extension principle applied to a function $f(x_1, \ldots, x_n)$ is defined as:

$$\begin{aligned} \mu_B(y) &= \sup_{\substack{x_1, \ldots, x_n \\ y = f(x_1, \ldots, x_n)}} \mu_A(x_1, \ldots, x_n) \\ &= \sup_{\substack{x_1, \ldots, x_n \\ y = f(x_1, \ldots, x_n)}} \min\left(\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\right) \end{aligned} \tag{4.29}$$

where $B$ is a fuzzy subset of the universe $Y$, such that $B = f(A_1, \ldots, A_n)$, with $A_1, \ldots, A_n$ fuzzy subsets of $X_1, \ldots, X_n$ respectively. Note that $A$

identifies the multi-dimensional set, obtained from the Cartesian product $A_1 \times \cdots \times A_n$, and so the second equality of Eq. (4.29) is a consequence of Eq. (4.9).

*Operations on fuzzy numbers*
The addition and scalar multiplication are defined according to Eq. (4.29). Assume to have two arbitrary fuzzy numbers defined as: $u = (\underline{u}, \bar{u})$, $v = (\underline{v}, \bar{v})$ and $k > 0$;

- Addition $(u + v)$ is defined as:

$$(u+v) = (\underline{u+v})(r), (\overline{u+v})(r) \tag{4.30}$$

with:

$$\begin{aligned} (\underline{u+v})(r) &= \underline{u}(r) + \underline{v}(r) \\ (\overline{u+v})(r) &= \overline{u}(r) + \overline{v}(r) \end{aligned} \tag{4.31}$$

- Multiplication by a scalar quantity is defined as:

$$ku = (\, (\underline{ku})(r), (\overline{ku})(r) \,)$$

with:

$$\begin{aligned} (\underline{ku})(r) &= k\underline{u}(r) \\ (\overline{ku})(r) &= k\overline{u}(r) \end{aligned} \tag{4.32}$$

Another important operation performed on fuzzy numbers is the comparison between a fuzzy number and a crisp (or scalar) number: it will assume an important role in Chapter 5 and therefore it is here introduced.

- Ranking functions
M. Adabitabar Firozja et al. [70] proposed a “ ranking function ” that is able to evaluate the possibility that a fuzzy number is greater (or smaller) than a crisp number.
Be $A$ a fuzzy number and $L$ the ranking function defined as:

$$L(A, \cdot) : \mathbb{R} \rightarrow [\,0, 1\,]$$

The ranking function is used to extend the natural ordering relation $\leq$ on real numbers to the ordering of fuzzy numbers with respect to

real numbers.
The possibility that the fuzzy number $A$ is less or equal to a crisp number $x$ is:

$$L(A,x) = \frac{\displaystyle\int_{-\infty}^{x} \mu_A(t)\,dt}{\displaystyle\int_{-\infty}^{+\infty} \mu_A(t)\,dt} \tag{4.33}$$

The ranking function $L(A,\cdot)$ is an increasing function (graph on the left of Figure 4.19): in fact, when $x$ is very small - say completely out of the number, on its left - the possibility that $A \leq x$ is null.
Dually, the possibility that $A \leq x$ when $x$ is completely out of $A$,

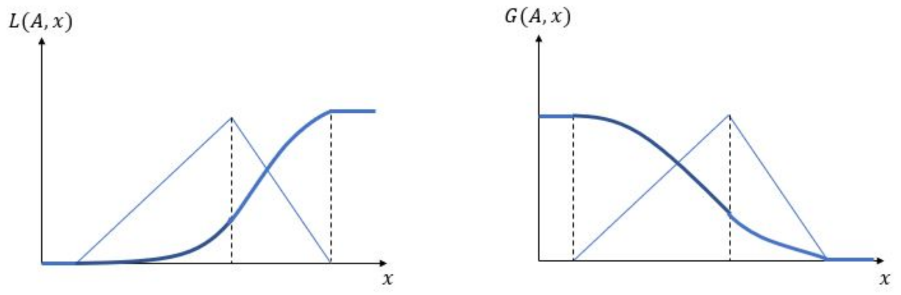


**Figure 4.19:** On the left: ranking function for ordering with respect to " $\leq$ "; on the right: raking function for ordering with respect to " $\geq$ "

on its right, is 1.
Similarly, another ranking function $G(A,\cdot)$ can be defined to compute the possibility that a fuzzy number $A$ is greater or equal to a crisp number $x$:

$$G(A,x) = \frac{\displaystyle\int_{x}^{+\infty} \mu_A(t)\,dt}{\displaystyle\int_{-\infty}^{+\infty} \mu_A(t)\,dt} \tag{4.34}$$

The ranking function $G(A,\cdot)$ is decreasing (graph on the right of Figure 4.19) for the dual reasons presented above.

Operations of subtraction between fuzzy numbers and multiplication/-division within fuzzy numbers are not generically defined, because they are not invertible operations. There are many theories and techniques to perform such operations, providing many additional assumptions on the fuzzy numbers involved. However, multiplication/division between fuzzy numbers won't be used by this work and therefore the explanation of such theories would be out of the scope of the thesis.

Concerning subtraction, many methods have been proposed in literature to compute it as distance between two fuzzy numbers.
The most relevant one is reported in the following.

*Hukuara-difference*
The Hukuara-difference has been defined [71] to overcome the situation in which, computing arithmetically the subtraction between two fuzzy numbers, the difference is not always invertible. Indeed, taking two fuzzy numbers $a$ and $b$, it may happen:

$$(a+b)-a \neq a$$

If a fuzzy number $c$ exists such that:

$$c+b=a$$

where "+" identifies a fuzzy summation, then $c$ is the Hukuara-difference of $a$ and $b$.
The condition to ensure that $c$ exists is that some translate of b is a fuzzy subset of a. Taking for instance $a=(1,3,5)$ and $b=(1,2,3)$, then:

$$(1,3,5)=(0,1,2)+(1,2,3)$$

where $c=(0,1,2)$ is called *H-difference* of $a$ and $b$.
Since it is not generally possible to guarantee the existance of $c$, in Section 5.2.2 an alternative method is proposed to counteract the restrictions imposed by such method.

## 4.6 Uncertainty propagation

The aim of this Section is to show the differences between a probability propagation process and a possibility one.
Consider a system for which the input parameters $y_1, y_2, \ldots, y_N$ are ill-known: only their probability distribution functions $f(y_1), f(y_2), \ldots, f(y_N)$ and their possibility distribution functions $\pi(y_1), \pi(y_2), \ldots, \pi(y_N)$ are given (Figure 4.20).
In the following, it will be shown how to propagate the uncertainty of the input parameters to the output of the system, both when considering a possibility and a probability framework.

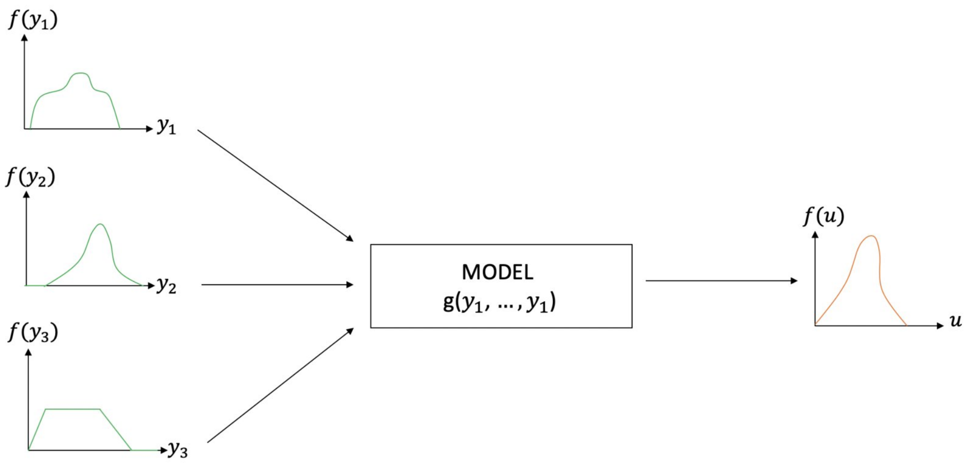


**Figure 4.20:** Propagation of the uncertainty through a multi-input system [64]

- **Probabilistic approach**: be $y_1$ and $y_2$ the not-precisely-known input parameters of the system, described by the probability distribution functions $f(y_1)$ and $f(y_2)$. Assume that:

$$\begin{aligned} g &= u(y_1, y_2) \\ Y_1 &\approx f_{Y_1}(y_1) \\ Y_2 &\approx f_{Y_2}(y_2) \end{aligned}$$

The uncertainty from $y_1$ and $y_2$ to the output of the system $u$, is propagated by making use of the canonical probability axioms, leading to:

$$\begin{aligned} F_U(u) &= P\{\,U \leq u\,\} = P\{\,g(Y_1, Y_2) \leq u\,\} = \\ &= \int_{(y_1,y_2)\,:\,g(y_1,y_2)\leq u)} f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)\, dy_1 dy_2 = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I_g(y_1, y_2) \cdot f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)\, dy_1 dy_2 \end{aligned} \tag{4.35}$$

with:

$$I_g(y_1, y_2) = \begin{cases} 1 & \text{iff } f(y_1, y_2) \leq u \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.36}$$

Since Eq. (4.35) involves infinite integrals, an exact solution might be difficult to find and therefore an approximated solution is often provided by means of alternative methods, such as the Monte Carlo method (MC). MC performs many simulations, to then come up with an estimate of the output of the system. Figure 4.21 [66] reports the basic mechanism of Monte Carlo approaches.

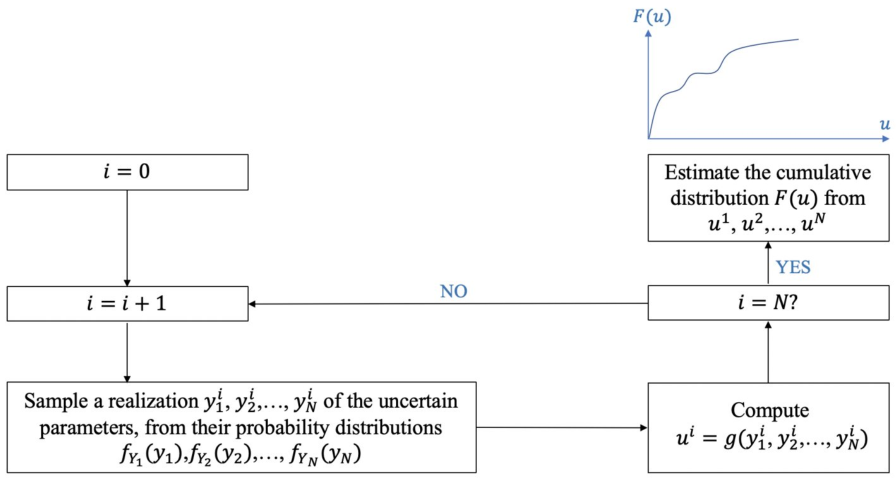

$F(u)$
$u$
$i = 0$
Estimate the cumulative distribution $F(u)$ from $u^1, u^2,\dots, u^N$
YES
NO
$i = i + 1$
$i = N?$
Sample a realization $y_1^i, y_2^i,\dots, y_N^i$ of the uncertain parameters, from their probability distributions $f_{Y_1}(y_1), f_{Y_2}(y_2),\dots, f_{Y_N}(y_N)$
Compute $u^i = g(y_1^i, y_2^i,\dots, y_N^i)$

**Figure 4.21:** Probabilistic propagation of the uncertainty, using Monte Carlo method [64]

First, the MC method generates N realizations (samples) of the random parameters of the system, according to their joint distribution. Each of these realizations defines a deterministic problem that is solved using a deterministic technique, generating a certain amount of data. Such data is lately combined through statistic, to estimate the response of the random system. Hence, the steps are:

– sample $(y_1^i, y_2^i)$ from $f_{Y_1}(y_1), f_{Y_2}(y_2)$
– corresponding award: $I_g(y_1^i, y_2^i)$

Consider $N$ trials $\{\, (y_1^1, y_2^1), (y_1^2, y_2^2), \dots, (y_1^N, y_2^N) \,\}$; then, the response $u$ of the system is characterized by:

$$F_U(u) = \frac{\sum\limits_{i=1,\dots,N} I_g(y_1^i, y_2^i)}{N}$$

For great values of $N$, the MC method can provide satisfactory results, describing well the statistical behaviour of the system [64]. However, the rate of convergence is quite slow, as it is proportional to the inverse of the square root of the number of samples collected, i.e. $\sim 1/\sqrt{N}$ . Therefore, if the processing time of a single sample is large, the slow rate of convergence makes MC a very time-consuming

method. For this reason, it might be unfeasible to perform simulations of complex models.
In particular, when objective information on the uncertainties is limited, the high computational cost of probabilistic analysis makes MC method not worthy, especially considering the fact that only approximated results (estimates) can be provided. For a practical example of this approach refer to Appendix.

- **Possibilistic approach**: consider first the SISO system with:
  - input parameter $Y$, that has values in $\mathbb{R}$ and its uncertainty is described by the possibility distribution $\pi_Y(y)$
  - output quantity $U = g(Y)$, with values in $\mathbb{R}$, whose uncertainty is represented by $\pi_U(u)$. It is determined using the formula of Eq.(4.28), which is derived from Zadeh's *extension principle* [63], (Eq. (4.29)).

$$\pi_U(u) = \sup_{y,\, g(y)=u} \left( \pi_Y(y) \right) \tag{4.37}$$

The *extension principle* is used to " extend " the transfer function of the system, $g$, from the $\mathbb{R}$ domain to the domain of the possibility distributions. The aim is to extend $g$ in order to work on it as if it is a possibility distribution. In this way the uncertainty (that is indeed represented as a possibility distribution) can be propagated to the output of the system. Overall, the main procedure consists of extending $g$:

  - from a function from $\mathbb{R}$ to $\mathbb{R}$
  - to a function from and to the class of all the possibility distributions defined on $\mathbb{R}$

The extension procedure passes through the following steps (Figure 4.22):

1. Select the set of $y$ values such that $g(y) = u$
2. Compute the corresponding set of $\pi_Y(y)$
3. Assign to $\pi_U(u)$ the maximum among the values of $\pi_Y(y)$ that were selected at point 2.

The outcome of point 3, $\pi_U(u)$, is the possibility distribution of the output of the system and, so, the representation of its uncertainty.

Consider now a multi-input system with:

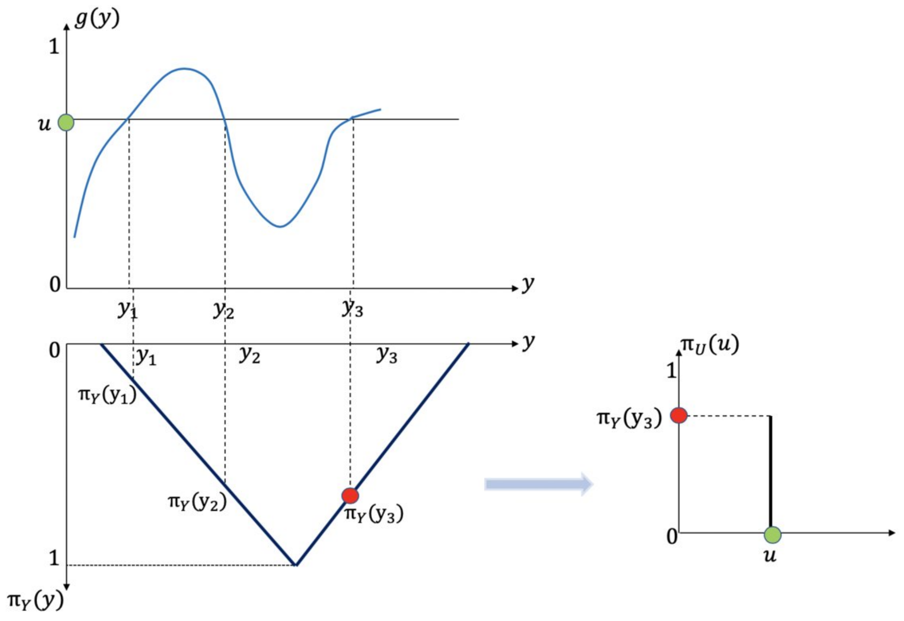


**Figure 4.22:** Propagation of the uncertainty with extension principle [64]

- input parameters $Y_1, Y_2, \dots, Y_N$ belonging to $\mathbb{R}$, whose uncertainty is described by the possibility distributions $\pi_{Y_1}(y_1)$, $\pi_{Y_2}(y_2)$, $\dots$, $\pi_{Y_N}(y_N)$
- output quantity $U = g(Y_1, Y_2, \dots, Y_N)$, whose uncertainty is given by $\pi_U(u)$. It is obtained from the *extension principle*, applying the formula in Eq. (4.28)

  $$\pi_U(u) = \sup_{y_1,y_2,\dots,y_n, g(y_1,y_2,\dots,y_n)=u} \pi_{Y_1,Y_2,\dots,Y_n}(y_1, y_2, \dots, y_n) \quad (4.38)$$

  where:

  $$\pi_{Y_1,Y_2,\dots,Y_n}(y_1, y_2, \dots, y_n) = \min\{\, \pi_{Y_1}(y_1), \pi_{Y_2}(y_2), \dots, \pi_{Y_n}(y_n)\,\} \quad (4.39)$$

  the above equation is obtained applying Eq. (4.10), related to the *Composition of fuzzy relations.*

As can be noticed, the propagation of uncertainty required by possibility distributions involves much easier computations.
In fact, Possibility Theory requires to apply simple equations (Eq. (4.38) and Eq. (4.39)), that do not involve integrals. In addition, it does not require to perform numerical approximations as MC methods. The essence

of the possibilistic approach is to start from approximated shapes (distributions) to then come up with a result that is yet approximated but outcome of a computationally reduced process.

In view of the given introduction about the approaches to deal with uncertainty, it is concluded that decision making - having the goal to account for the influence of uncertainty on choices - has raised the need of quantitative methods to deal with the above-mentioned types of uncertainty.
Probabilistic approaches quantify uncertainty by probabilities and «are recommended when background knowledge is strong enough to allow uncertainty be given in terms of likelihoods or degrees of belief» [69]. When uncertainty expresses the lack of knowledge, and so the background knowledge is poor, it is better to give uncertainty a non-probabilistic treatment. For instance, uncertainty quantified as fuzzy numbers (4.5.2), i.e. possibility distributions, is a non-probabilistic and quantitative approach.
Note that by “ handled ” it is intended quantification, based on given background knowledge, and its propagation in the considered model.
In conclusion, both theories can be suitable to manage an uncertainty propagation problem. One approach doesn't exclude the other (as can be seen in the example below). Nevertheless, if the available information is scarce, it is better to use Possibility Theory. This thesis will assume such worst case scenario, proving that good results are still obtained.

*Example: coexistence of possibility and probability*
To illustrate the difference between probability and possibility, a simple scenario is given: consider the statement “ The robot machines $X$ pieces in a day ”, where $X$ takes values in a space of values $U = u$.
The possibility distribution can be associated with $X$ by interpreting $\pi_X(u)$ as the degree of ease with which the robot can machine $X$ pieces in a day. In addition, a probability distribution can be associated with $X$ by interpreting $P_X(u)$ as the probability that the robot machines $u$ pieces in a day.
The values of $\pi_X(u)$ are determined basing on some implicit/explicit criteria, that assess the degree of ease with which the robot machines pieces. As an example, suppose that for a specific application the time taken to machine a single piece is very high, and so it is known that no more than $Q$ pieces can be machined in a day. This source of knowledge acts as a constraint and therefore is a criteria that imposes physical limits on the possibility distribution.
Concerning probability, the values of $P_X(u)$ are determined studying the

past data statistically: knowing how many pieces the robot had machined in the past, considering $N$ experiments, a value of " probability that the robot will machine $X$ " can be computed in a stochastic way.
The values of $\pi_X(u)$ and $P_X(u)$ might be as shown in table (4.1).

| u | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\pi_X(u)$ | 1 | 1 | 1 | 1 | 0.8 | 0.6 | 0.4 | 0.2 |
| $P_X(u)$ | 0.1 | 0.8 | 0.1 | 0 | 0 | 0 | 0 | 0 |

**Table 4.1:** Probability and possibility distributions associated with X

It can be observed that, whereas the possibility that the robot machines 3 pieces in a day is 1, the probability of the same event is quite small: 0.1. This behaviour reflects the " consistency principle " of Zadeh, for which «a high degree of possibility does not imply a high degree of probability, nor does a low degree of probability imply a low degree of possibility. However, if an even is impossible, it is bound to be improbable» (refer to [63]). Therefore, possibility and probability are not bound to be proportional: they can be very different and not strictly related.

## 4.7 Relations between probability and possibility measures

Let's consider a fuzzy set $A$ having a possibility distribution $\pi(y)$. The possibility that a particular element $y$ of $U$ belongs to $A$ can be evaluated.
For instance, consider $y$ to be comprehended in the finite interval $I = [y_1, y_2]$, (Figure 4.23).

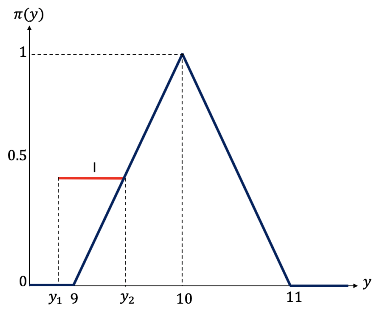


**Figure 4.23:** Possibility measure of a finite interval [64]

Then:

- the *possibility measure* of the set I, $\Pi(I)$, can be computed as in Eq. (4.40): it takes the values of the possibility distribution of $A$, for every $y$ belonging to $I$.

$$\forall I \subseteq X, \qquad \Pi(I) = \max_{y \in I} \pi(y) \tag{4.40}$$

  where $\pi(y) = \Pi(\{ y \})$.

- The *necessity measure* $N(I)$ is related to the possibility measure $\Pi(I)$ by (4.41):

$$\forall I \subseteq X, \qquad N(I) = 1 - \Pi(\bar{I}) = 1 - \max_{y \notin I} \pi(y) \tag{4.41}$$

  where $\bar{I} = U - I$.

Eq. (4.41) is a numerical expression for a duality relationship: an event is necessary when its contrary is impossible [65]. From what stated above, the following definition (4.42) is derived:

$$1 - \max_{y \notin I} \{ \pi(y) \} \leq Poss\{ y \in I \} \leq \max_{y \in I} \{ \pi(y) \} \tag{4.42}$$

In other words, the possibility that an ill-known variable $y$ belongs to an interval $I$ is comprehended between the *necessity measure* and the *possibility measure*.
In addition, the necessity measure corresponds to the “ lower limiting probability value ” and the possibility measure to the “ upper limiting probability value ” [64]:

$$\begin{array}{c}\text{Lower limiting}\\ \text{probability value}\end{array} \leq P\{ y \in I \} \leq \begin{array}{c}\text{Upper limiting}\\ \text{probability value}\end{array} \tag{4.43}$$

In fact, a high degree of possibility does not imply a high degree of probability, nor does a low degree of probability imply a low degree of possibility.
This concept was formalized by Zadeh through the “ consistency principle ” [63], which allows to define the degree of consistency between a probability distribution and a possibility distribution:
Suppose that a variable $X$ can take values $u_1, u_2, \ldots, u_N$, with probability $P = (p_1, \ldots, p_n)$ and possibility $\Pi = (\pi_1, \ldots, \pi_n)$ respectively; then, the

degree of consistency between the probability and possibility distributions is:

$$C_z = \sum_{i=1}^{n} \pi_i \cdot p_i \tag{4.44}$$

Zadeh made clear that the probability-possibility consistency is not a precise law or a relationship that is intrinsic in the concept of possibility and probability distributions. It is instead an approximate formalization of the fact that decreasing the possibility of an event leads to reduce also its probability, but not vice-versa [63].
Considering the consistency principle (4.44) and the definitions of possibility and necessity measure (4.41), it can be stated that:

$$\begin{array}{ccc} \text{Necessity measure} & & \text{Possibility measure} \\ \text{N(I)} & & \Pi(I) \\ \downarrow & & \downarrow \\ \begin{array}{c}\text{Lower limiting}\\ \text{probability value}\end{array} & \leq P\{\, y \in I \,\} \leq & \begin{array}{c}\text{Upper limiting}\\ \text{probability value}\end{array} \end{array} \tag{4.45}$$

Note that, even when an infinite set I $= [-\infty, u]$ is considered, the possibility that $y$ is smaller than a generic $u$ of the universe of discourse $U$ ($Poss\{\, y \leq u \,\}$) is still comprehended between two limiting cumulative distributions (Figure 4.24).

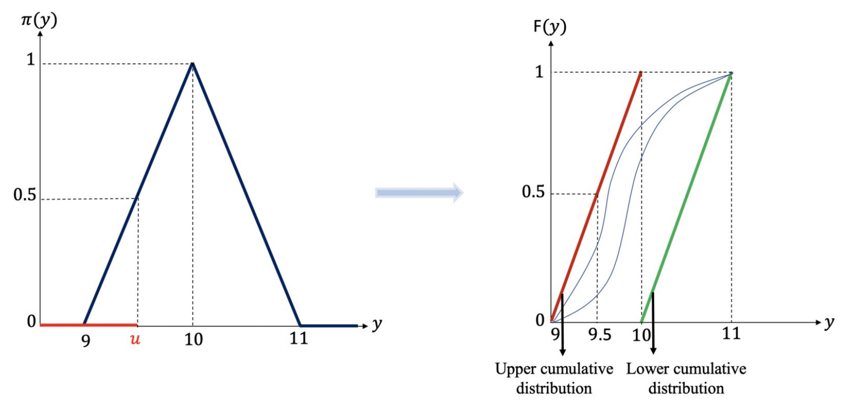


**Figure 4.24:** Possibility distribution comprehended between lower and upper cumulative distributions [64]

These cumulative functions are computed using the same formulas of the case in which I was a finite set (4.42), with the only difference that instead of evaluating $Poss\{y \in I\}$, $Poss\{y \leq u\}$ is considered:

$$1 - \max_{y \notin I} \{ \pi(y) \} \leq P\{ y \leq u \} \leq \max_{y \in I} \{ \pi(y) \} \tag{4.46}$$

Therefore,

$$1 - \max_{y \notin I} \{ \pi(y) \} \leq F(u) \leq \max_{y \in I} \{ \pi(y) \} \tag{4.47}$$

The green line in Figure 4.24 represents the lower limiting probability; the red line represents the upper limiting probability. As a consequence of Eq. (4.47), the area comprehended between the green and red lines is the region in which all the plausible probability distributions can lay (Figure 4.25).

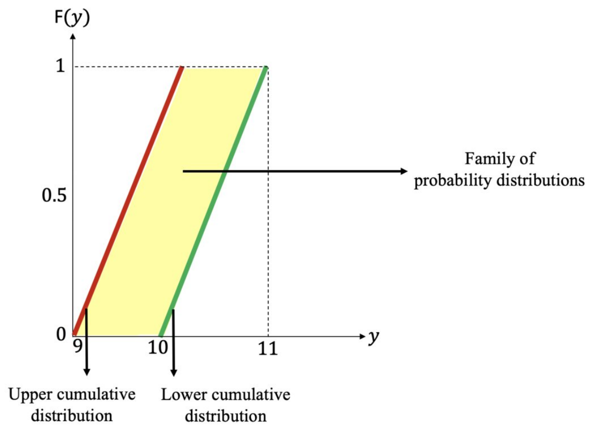


**Figure 4.25:** Family of probability distributions between lower and upper cumulative distributions [64]

# Chapter 5

# Scheduling of robotic activities with Fuzzy Theory

In this Chapter it will be shown how the theory presented in Chapter 4 has been adapted and used in this thesis; in particular, it will be illustrated how Fuzzy Theory is used to obtain a scheduling algorithm capable of accounting for the uncertainty embedded in the duration of tasks.
To account for such uncertainty, the scheduler generates reachability trees, created with a method explained in this Chapter, and the uncertainty is taken into consideration by associating it to temporal transitions, which are represented as possibility distributions.

In Section 5.1 it is explained how the information about duration of tasks is represented. In particular, the method used to create possibility distributions starting from collected samples is introduced, giving a practical explanation on triangular fuzzy numbers that are created and associated to duration of tasks.
Then, an overview on the predictive algorithm is given; it is used in this thesis to acquire information related to the human operator. Section 5.2 shows the procedure to generate the reachability tree, presenting in detail how the possibility is assigned to each of its nodes.
In Section 5.3 the way to find the optimal plan, which brings to the best performance for the system, is pointed out. To this end, the MPC approach and the concept of cost function are presented. Eventually, the pseudo-code of the developed scheduling algorithm is reported to summarize the whole process.

## 5.1 Duration of actions: triangular fuzzy numbers

In recent decades the use of Fuzzy Theory in engineering has increased significantly. Fuzzy science is able to construct models which can process qualitative information almost like a human.
In fact, in this thesis, the information available is assumed to be of qualitative type: the data describing duration of human and robot actions is affected by an uncertainty that cannot be associated to any particular probabilistic distribution.
The action of a robot arm can be interrupted or slowed down to avoid collision with a human or with the other robot arm: at every execution of the algorithm, the duration of such action might vary and the way it varies is not predictable.
Moreover, if the aim would have been to use probabilistic representation, first the shape of the distribution must have been assumed. Then, $N$ experiments must would have been performed, collecting at every trial $X$ samples: the greater the $N$, the more accurate is the probabilistic distribution associated to the considered duration.

To avoid performing a big amount of preliminary experiments, an alternative method is used in this thesis: a unique set of samples is collected and converted into possibility distribution.
Then, such set of samples is updated at every iteration of the algorithm, according to new information retrieved from the real system.

As anticipated, the decision-making problem handled in this thesis finds its basis on fuzzy numbers.
Recalling the definitions of Chapter 4, a fuzzy number is a particular case of a fuzzy set: it must be a convex and normal set, with membership function varying between 0 and 1.
The function itself can be an arbitrary curve whose shape is chosen according to requirements of simplicity and efficiency.
In this thesis, the time associated to any human or robot action is described as a fuzzy number (or at most as a fuzzy singleton if its value is crisp) instead of as a fuzzy set.
Such constraint is due to the need of performing mathematical operations, that would otherwise be impossible to perform if the shapes were not convex. In fact, as will be explained, to propagate the uncertainty along the reachability tree, it is required to execute algebraic operations, as

summation between fuzzy numbers.

To perform a fuzzy analysis, it is first needed to create fuzzy numbers through a process called "fuzzification": it converts an uncertain quantity into a fuzzy value, using membership functions.

In the following it will be shown how fuzzy numbers are created starting from samples.
First of all, two different ways are used to collect samples. For robot actions, at the first invocation of the algorithm, their duration is measured experimentally, to create a first rough representation of such duration. Then, iteration by iteration, these distributions are overwritten by a mechanism that learns the behaviour of the robot, substituting the above mentioned time distributions with more accurate ones.
The distributions that represent the duration of human actions are built using samples that are provided by a predictive algorithm, which is out of the scope of this thesis but is shortly presented in the following.

### 5.1.1 Predictive algorithm

The human, interacting with a robot in a collaborative context, is an agent whose actions are uncontrollable and can only be predicted.
The predictive algorithm developed in [51] is exploited in this work. It is able to give the robot the knowledge about the human with which it is co-operating, to better adapt to his/her needs.
In fact, such prediction is used by the scheduling algorithm that controls the robot, in order to better assist the human operator and, at the same time, be productive.

The algorithm [51] predicts the possible starting time instant of the Human Actions or Human to Robot Actions, starting from a current time instant, within a certain temporal horizon.

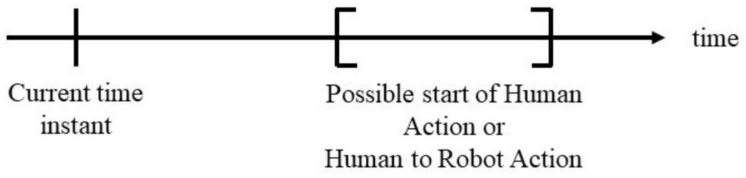


**Figure 5.1:** Predictive algorithm [73]

The algorithm provides possible samples corresponding to the time instants in which the human operator is expected to start an action. These samples are considered as qualitative information, an estimate coming from an expert, and so they are tackled with possibility theory, representing them as fuzzy triangles.

### 5.1.2 Creation of possibility distributions

The samples related to the robot and human actions are updated at each iteration of the algorithm and manipulated by a method that transforms them into possibility distributions.
As anticipated, the possibility distributions used in this thesis have triangular shape, because such shapes are able to describe in a good way the possibility associated to events.
Recalling the notation of triangles presented in Chapter 4, a triangular number is denoted as $(a, b, c)$, where $a$ and $c$ have zero possibility, $b$ has unitary possibility and the values between $a$ and $c$ have membership grade $\in (0, 1)$.

Given a set of samples, many methods exist to assign the corresponding membership degree, which basically consists of finding the vertices $a$, $b$ and $c$ of the triangle that better represents such samples.
Many methods have been proposed in literature to choose the appropriate membership generation technique. There is no objective way to evaluate the goodness or correctness of such methods and, mostly, the choice of the method depends on the kind of problem to handle and on the type of data that are available.

The only necessary condition that has to be satisfied to guarantee that the method is "correct" is to satisfy Zadeh's consistency principle: given a set of data and deriving probability and possibility distributions from it, they must provide a sufficiently positive consistency degree (see Section 4.7). This is to ensure that a lessening of the possibility of an event tends to lessen its probability (Eq. (4.44)).

In the following, two different methods for building the possibility distributions will be compared, considering some sets of samples obtained from random variables distributed as Gaussian Mixture.
Notice that here the aim is to compare the consistency of methods, assuming the samples to be generated from probability distributions.
The latter assumption is not anyway necessary considering a generic set of

samples.

Two methods have been tested on Matlab, to verify which one would provide the most realistic triangular distribution, to choose such method as a general one to use in this thesis.
The two methods that have been compared are:

- *Method 1: symmetric method*
  The first method consists of constructing a triangle that has most possible value $b$ coincident with the mean value $M$ of the normal distribution of data, whose vertices $a$ and $c$ are computed as $(M - \sigma)$ and $(M + \sigma)$ respectively, where $\sigma$ is the standard deviation [72].
  Nevertheless, this method builds only triangles that are symmetrically distributed around the mean point $M$, and so does not truly model the duration of tasks because, in a real situation, the data are not generally homogeneously distributed around a unique point.
  It would be rather accurate to use a method that creates scalene triangles instead, adapting to the variability of samples and so providing different triangular shapes according to the case of interest.

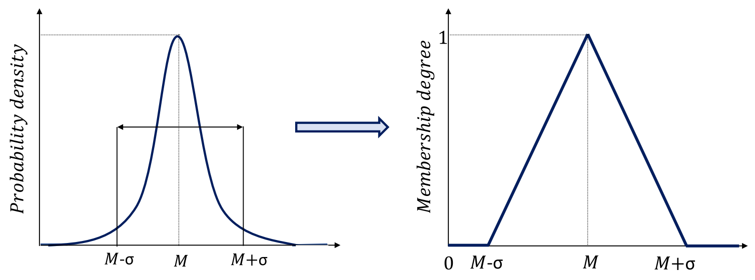


**Figure 5.2:** Symmetric method [72]

- *Method 2: asymmetric method*
  The method proposed by Amin Amini and Navid Nikraz [72] suggests a procedure that, starting from statistical data and frequency charts, constructs non-isosceles triangular fuzzy numbers. This method associates membership degree 1 to the mean value of frequency chart, and determines the membership degree of the other points distributing the standard deviation around the mean value.
  The final triangle will be formed through the following steps:

- Computing the average value $M$ of frequency chart data and standard deviation of data, $\sigma$;

- Finding the continuous function $f(x)$ that better describes the frequency chart, in which $X$ is the axis of scale degrees and $Y$ is the axis indicating the frequency data;

- Introducing and computing the following parameters, assuming that the $X$-axis is graded from 0 to $K$, where $K$ is the value of the highest sample and $\sigma_L$, $\sigma_R$ are the re-distributed standard deviations:

$$L_M = \int_0^M f(x)\,dx \qquad \sigma_L(M) = \frac{\sigma \cdot S}{(1+S)}$$
$$R_M = \int_M^K f(x)\,dx \qquad \sigma_R(M) = \frac{\sigma}{(1+S)} \tag{5.1}$$
$$S = \frac{L_M}{R_M}$$

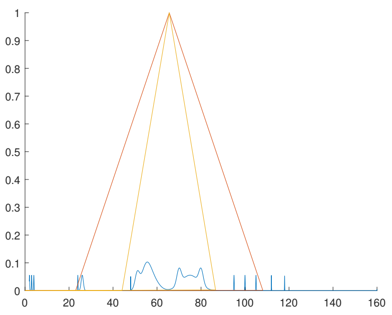
- Finding lower limit (LL) and upper limit (UL) as:

$$LL = M - \sigma_L(M)$$
$$UL = M + \sigma_R(M)$$

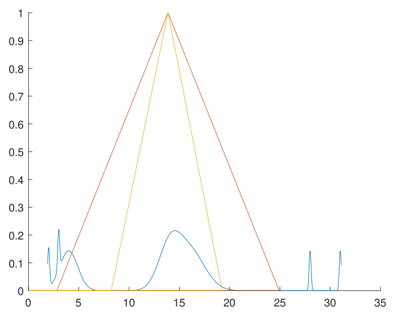
- Scaling the data in the form of fuzzy number membership function, assigning membership degree 1 to the mean value and membership degree 0 to UL and LL.

Referring to Eq. (5.1), $L_M$ is the area under the frequency graph for the left side of $M$ and $R_M$ is the area under this diagram on the right side of the mean point.
The values $\sigma_L(M)$ and $\sigma_R(M)$ are obtained distributing the standard deviation with respect to the ratio $S$, using a direct proportion.
By doing so, the region with bigger area due to more scattered responses leads to a bigger boundary around the average value, which represents higher uncertainty, whilst the area in which the samples are more concentrated leads to smaller boundary and so less uncertainty.

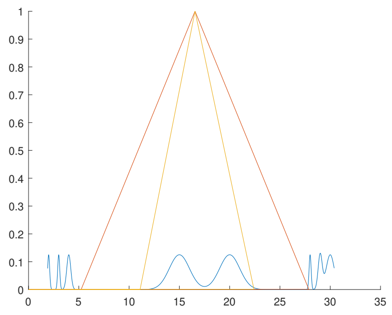
Figure 5.3 reports the results obtained by the two aforementioned methods. The blue lines identify the initial probability distribution generating



**(a)**



**(b)**



**(c)**

**Figure 5.3:** Triangular distributions: symmetric and asymmetric method with sets of samples generated by different distributions

the samples (mixture of Gaussian distributions), the red ones are the triangular fuzzy numbers obtained from the symmetric method and the yellow ones are obtained with the asymmetric method.
It can be noticed that, even though the points of Figure 5.3a are more concentrated between 40 and 80, the symmetric method built a triangle such that also the points that are quite far (close to 20 and around 100-120) are incorporated.
The same scenario holds for Figure 5.3b and Figure 5.3c, where the symmetric method provides triangles that try to include as many samples as possible, even though they might be very far from the most dense area.
On the other side, the asymmetric method builds triangles that are mostly concentrated on the region where the samples are more concentrated, excluding the few ones that deviate from it.
Using both methods the consistency coefficient is high (Table 5.1) which proves that they can both be used. It is important to remark that the

| Test | a) | b) | c) |
|---|---|---|---|
| $C_{consistency\_symmetric}$ | 40.9893 | 138.7745 | 87.1718 |
| $C_{consistency\_asymmetric}$ | 28.3794 | 117.4175 | 60.2389 |

**Table 5.1:** Consistency coefficients with symmetric and asymmetric method, considering Examples of Figure 5.3

aim is not to choose the method that provides the highest coefficient, but a method that satisfies the consistency principle and, at the same time, adapts to the variability of situations.
Hence, the major task of this thesis is to account for big amounts of uncertainty, i.e. considerable variations in duration of tasks: the samples describing the duration of actions are expected to be very scattered, as in Figure 5.3b.
Thus, a method that provides triangles capable to adapt to different sets of samples, and not only samples concentrated around a mean point, is needed: the asymmetric one has been judged to be the most suitable and general method and, therefore, is the one used throughout this thesis.

## 5.2 Computation of the fuzzy reachability tree

The aim of this Section is to show how the reachability tree is generated by the scheduling algorithm, pointing out its structure and how possibility is associated to it.

### 5.2.1 Reachability tree: set of nodes

Once the FTPN that models the system has been defined (see Section 3.2), it is possible to generate the reachability tree. First of all, a temporal horizon is established, i.e. a time limit over which it is not of interest to know how the system evolves. Then, all the possible ways in which the system can evolve, from the actual state within the scheduling horizon, are computed.

The reachability tree is a set of nodes, where each node is a tuple representing a specific state of the system. In general, a node is described by the following elements:

- `Reached_Marking`: represents the state reached by the system (see Section 3.1.1). It allows to know if agents or resources are available in that specifc state, or if and which actions are being executed in that particular state.
  The initial state of the FTPN is represented by the *root node*, which is the first node of the reachability tree.

- `Father_Node`: every node has a " father node " from which is generated, i.e. the node that precedes it in the reachability tree. This parameter contains the marking of the father node.

- `Enabled_Transitions`: keeps track of the transitions that are enabled in the considered state of the system (see Section 3.1.1), allowing to predict the possible evolution of the system, starting from that specific state.

- `Children_List`: is the set of nodes that are generated by the transitions that are enabled in the considered node and that are truly reachable. Only the reachable nodes are added to the reachability tree (as it will be later explained, not all the nodes that can be generated by the enabled transitions can effectively be reached, because some of these nodes would have an arrival time incompatible with the system evolution and so their transition wouldn't be allowed to fire). The nodes contained in this set will be from here on called *children nodes*.

- `Generator_Transition`: is the transition that generates the considered node, i.e. the transition that links the father node to the current node.

- `Arrival_Time`: represents the predicted time needed to reach the considered node. For the root node, it is 0; for a generic node, it is represented by a triangular fuzzy number (for its computation refer to Section 5.2.3).

- `Possibility_To_Reach_The_Node`: it is the possibility to reach the current node. Every node of the reachability tree has its own possibility to be reached, depending on the amount of uncertainty characterizing the duration of actions on the specific branch; in Section 5.2.4, it will be explained the method to compute such parameter. In general, it is a real value comprehended between 0 and 1 and is computed considering the *Possibility to reach the node* of the predecessor nodes on the same branch of the tree.

- ***Robot_Waiting_Coefficient***: it is a coefficient expressing the amount of time for which the robot has remained inactive. This coefficient is used in the computation of the cost function, to select the best branch of the reachability tree (see Section 5.3.2).
  In general, it coincides with the number of times for which the transition of type " Robot waits " occurred: such transition corresponds to commanding the robot to wait.
  One might wonder why should a transition of type " Robot waits " be used in a FTPN. Although it seems that it adds waiting time, and so that it increases the robot waiting coefficient, reducing the performances, it might instead be used to favour the human, saving in terms of human waiting time.
  In fact, it might happen that the robot is inactive and its enabled transitions, that the scheduler can choose, are:

  - Start a Robot Action, with duration of 10 seconds
  - Wait for 1 second

  At the same time, the predictive algorithm informs that the human will finish his/her current action in 2 seconds and that then might want to perform a Human Action for which he/she will need a piece machined by the robot, so a piece coming from a Robot to Human Action.
  Therefore, if the scheduling algorithm chooses to let the robot perform the Robot Action, then the human would have to wait for at least

  $$(10 - 2)s + \text{duration of Robot to Human Action}$$

  before starting the intended Human Action.
  On the other side, if the scheduling algorithm chooses to let the robot wait for 1 second, it gives the robot the opportunity to start the Robot to Human Action after 1 second, instead of after 10 seconds. In this way, the robot waiting time is slightly increased but the human waiting time is significantly reduced.

- ***Human_Waiting_Coefficient***: it is a coefficient proportional to the time for which the human has been waiting until such node was reached. As the previous coefficient, it is used in the cost function.

- ***J_coefficient***: is the parameter used to choose the best branch of the tree, and so the best commands for the robot (see Section 5.3.3). It considers the waiting time of agents in order to optimize it and

also takes into account the possibility to cover the chosen branch.
In fact, the reachability tree is composed of many branches that represent all the possible developments of the system: each of these paths is affected by uncertainty and therefore described by possibility. Paths that are very uncertain would have a very low possibility to be chased.
Thus, a branch that guarantees very convenient waiting times but that has low possibility to be chased would be excluded from the set of branches that can be chosen as best ones.

A detailed presentation on how the above parameters are used and computed is provided in the following.

### 5.2.2 Generation of the reachability tree

Starting from the current state of the system, i.e. the root node of the reachability tree, the algorithm generates the list of all the states that can be reached within the scheduling horizon. Every node of the reachability tree is reached by firing a transition enabled in the state corresponding to the father of that node. In other words, starting from the root node, the algorithm generates all its *children nodes*, i.e. all the nodes that can be reached by the current state and that represent the ways in which the system can evolve.
The root node is inserted in a list of *Unexplored nodes*, that contains the nodes that have been generated and have still not been analyzed.
The algorithm follows an iterative procedure: it analyzes one by one the unexplored nodes, to check if, starting from each one of them, the system can evolve to some other states.
If this holds, those states are said to be *reachable* and the respective nodes are added to the tree, to the list of unexplored nodes and to the children list of the correspondent father node.
The expansion of a branch is interrupted whenever the arrival time in a node is over the temporal horizon or if the analyzed node is generated by a Human Action or Human to Robot Action (see Section 3.3). Indeed, if the human operator performs an action, the token of the place of the FTPN related to the availability of the human is removed. Since the human is considered to be an external agent not controlled by the FTPN, the token related to its availability is never brought back by the natural evolution of the FTPN.
Such token is re-inserted in the place only when an external sensor (camera) outputs the information that the human has finished his/her action.

For this reason, it would be useless to keep on expanding the branch along which the transition Human action or Human to Robot action has fired: enlarging the branch would simulate a situation in which the place related to the human availability would never be filled with a token again, and so the human would not be working, increasing the costs related to his/her waiting time.
The nodes that do not generate any other children nodes are called "leaf nodes". When a leaf node is reached, the *Human waiting coefficient* is computed as follows: it is proportional to the time for which the human operator has been inactive, from the initial to the current state. It is computed as the temporal distance between the time in which the transition that enables the human to wait fired and the time in which the Human Action was executed. Since the time related to the Human Action is represented as possibility distribution, the above mentioned procedure corresponds to calculating the difference between two fuzzy numbers. Nevertheless, the " distance operation " is not generally mathematically defined for fuzzy numbers, as anticipated in 4, and therefore an alternative method will be presented.

### Alternative method to evaluate fuzzy distance

Consider, for example, two fuzzy numbers: $a = [1, 4, 6]$, $b = [5, 7, 12]$, as represented in Figure 5.4.

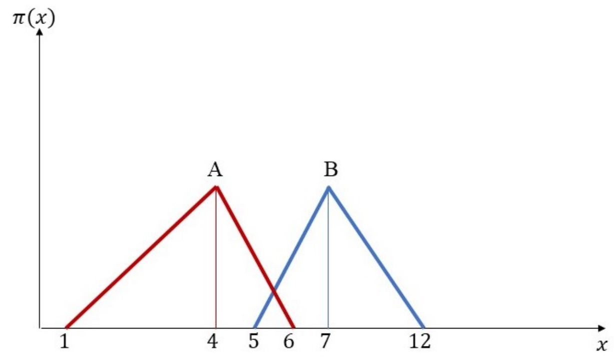


**Figure 5.4:** Evaluation of distance between two fuzzy numbers

The distance of $b$ from $a$ is the fuzzy number $c$ such that:

$$b = a + c$$

Therefore,

$$c = [3, 4, 3]$$

which is not a unimodal number and so cannot be accepted as a fuzzy number. This simple example shows that the subtraction, or Hukuare-difference (see Section 4.5.2), is not always defined.
In this case, in fact, the subtraction is not an invertible operation:

$$(a+b)-a \neq a$$

From this, the need to find a generic method that can always be applied, irrespective of the specific dimensions of the triangles being compared.

Recalling that the exploration of the reachability tree stops whenever a Human Action or Human to Robot Action starts, assume that the system reaches a state identified by node $P_i$ and that such node was generated by a Human Action: the tree is not developed anymore on this branch and the *Human waiting coefficient* is computed.
Before starting a Human Action, the human expresses the intention to perform the related action: the time elapsed between the intention and the actual start is referred to as human waiting time. To evaluate the human waiting time, the nodes preceding $P_i$ on the same branch must be searched: one of those, $P_k$, is the one created by the transition that enabled the human to wait.
The mathematical value of the human waiting time is normally expressed by the “distance” between the arrival time in $P_k$ ($a = [a_1, a_2, a_3]$) and the arrival time in $P_i$ ($b = [b_1, b_2, b_3]$).
The method that alternatively expresses such distance provides a coefficient that is proportional to the human waiting time and that allows to coherently evaluate the *Human waiting coefficient* of all the nodes of the reachability tree, allowing to compare them.

$$K_{wh} = \varphi_1 \cdot \varphi_2 \cdot \varphi_3 \cdot \varphi_4 \tag{5.2}$$

In the following, the interpretation of the above equation is given.
A simple method would measure the distance between two triangles considering either a worst case scenario – and so measuring the distance between the least-right possible value of $B$ ($b_3$) and the least-left possible value of $A$ ($a_1$) – or it would consider the highest-possible case scenario, i.e. measuring the distance between the two most possible values, $a_2$ and $b_2$.
Nevertheless, it is easily noticeable that these measures are not precise enough.
In fact, looking at Figure 5.5, where the peaks are the same in both cases, in one case the two triangles have an intersected area whilst in the second case they are not intersected. In the latter, the possibility to reach $P_i$ in

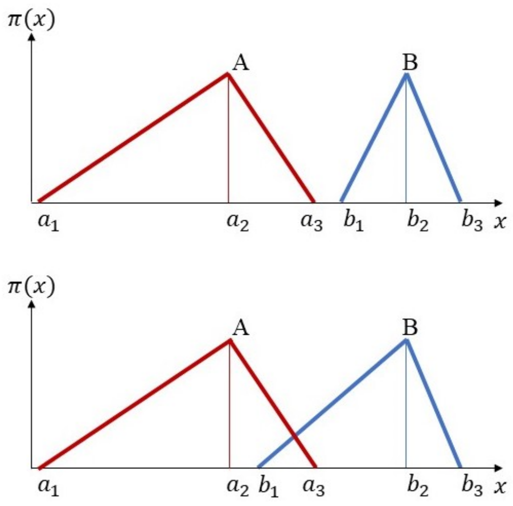


**Figure 5.5:** $\varphi_1$ in human waiting coefficient: case 1 (top) and case 2 (bottom)

a time instant $\in [\,b_1, b_3\,]$ very close to $[\,a_1, a_3\,]$ is less than it would be in the first case: in the second case, $P_i$ can be reached at most at $a_3$, which precedes $b_1$, whilst in the first case $a_3$ is after $b_1$ and so $P_i$ can be reached in a time instant possibly closer or inside the support of $B$. Therefore, the possibility to reach $P_k$ in a time instant close to the instant in which $P_i$ is reached is higher in case 1, which then is the case in which the waiting time is smaller.
Thus, the need for a term that reduces the waiting coefficient as the intersected area increases:

$$\varphi_1 = 1 - \psi$$

where $\psi \in [\,0, 1\,]$ indicates the result of the normalization of the intersected area, performed to make it range between 0 and 1. To do so, the intersected area has been divided by the maximum possible intersected area, which is a triangle of height 1 (all the triangles that are fuzzy numbers, i.e. the ones considered in this thesis, have height 1) and support equal to the temporal horizon considered for the scheduling.
In this way the triangles representing the arrival times of all the nodes of the reachability tree are comparable.
If the intersected area is null, $\varphi_1$ is maximum and so will “ maximize ” the human waiting time (not maximize in absolute terms, because other terms of Eq. (5.2) might reduce $K_{wh}$).
If the intersected area is maximum, the two triangles are overlaid and therefore the coefficient of human wait is zero. As the area increases, $K_{wh}$ decreases.

The intersected area cannot be the only parameter to be considered. As represented in Figure 5.6, two triangles with same support but different peaks should influence in a different way $K_{wh}$.

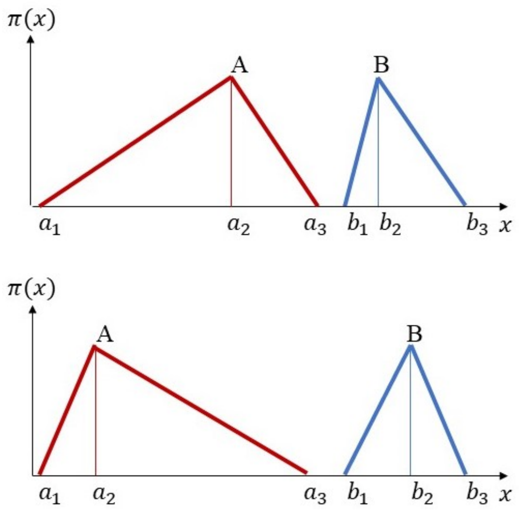


**Figure 5.6:** $\varphi_2$ in human waiting coefficient: case 1 (top) and case 2 (bottom)

In fact, in case 1 peaks are much closer than in case 2, meaning that the most possible value of distance between $A$ and $B$, i.e. $b_2 - a_2$, is smaller in case 1. The same holds if the two triangles are intersected. From here, the need to include $\varphi_2$ in the formula of Eq. (5.2):

$$\varphi_2 = 1 + \gamma$$

where $\gamma$ indicates the normalization of the distance $b_2 - a_2$ over the maximum distance possible, which is the temporal horizon.

- if $b_2 - a_2 > 0$, $B$ has its peak after the one of $A$; the more $b_2$ is on the right side of $a_2$, the more plausible is that the waiting time increases. In fact, in this case $\gamma \in (\,0, 1\,]$ and so $\varphi_2 \in [\,1, 2\,)$, which makes $K_{wh}$ increase.

- if $b_2 - a_2 < 0$, $B$ has its peak before the one of $A$: it is more possible that $P_i$ is reached at a time instant $\in [\,b_1, b_3\,]$ very close to the time instant $\in [\,a_1, a_3\,]$ in which $P_k$ is reached. This translates in a lowered waiting time and, in fact, $\gamma \in [\,-1, 0\,)$ which leads to $\varphi_2 \in [\,0, 1\,)$ and so to a reduction in $K_{wh}$.

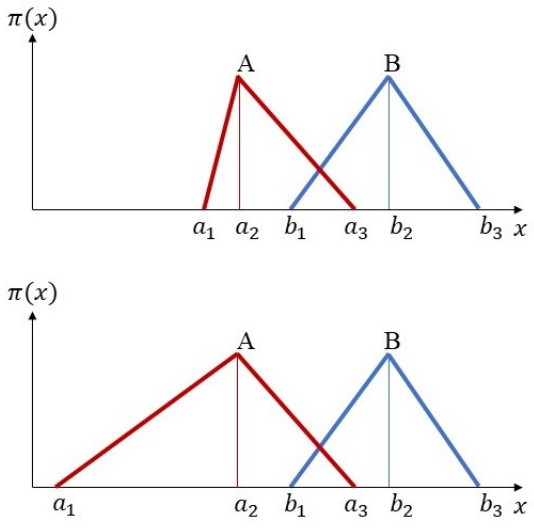


**Figure 5.7:** $\varphi_3$ in human waiting coefficient: case 1 (top) and case 2 (bottom)

Moreover, consider the case of Figure 5.7, where peaks and intersected areas do not vary and the left vertex of $A$ does. It can be noted that the possible values of $A$ with which $P_k$ can be reached are more developed on the left side in case 2: this means that the possibility to reach $P_k$ in an instant $\in [\, a_1, a_3 \,]$ that is further from $[\, b_1, b_3 \,]$ is higher in 2 rather than in 1.
In other words, the region in which only $A$ has values greater than zero (region with support $b_1 - a_1$) is wider in case 2 and so the possible human wait is higher in case 2.
Thus, a proportional term is needed to consider how long is the support $b_1 - a_1$: if it increases, the coefficient proportional to the waiting time should increase as well:

$$\varphi_3 = 1 + \eta$$

where $\eta$ indicates the normalization of the distance $b_1 - a_1$ over the maximum distance possible.
Note that, since $A$ is the arrival time of $P_k$, which is an ancestor of $P_i$, and because of how time is treated in this thesis (see Section 5.2.3, “Phase 1”), $\eta \in [\, 0, 1 \,]$ and can never be negative.

The dual case is represented in Figure 5.8, where intersected area, peaks and left vertices do not vary: by varying the right vertices in can be noticed that $B$ can assume values much more scattered and further from $A$ (case 2). In this case, the possible waiting time is greater than in case 1, so

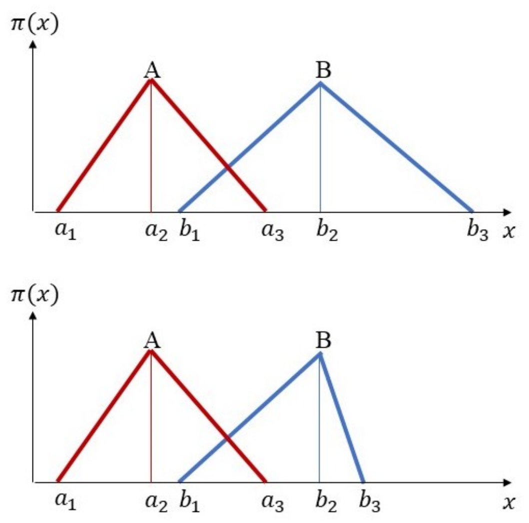


**Figure 5.8:** $\varphi_4$ in human waiting coefficient: case 1 (top) and case 2 (bottom)

$K_{wh}$ should also consider the length of the support in which $B$ has values completely out of the distribution $A$. The wider it is, the most plausible is that node $P_i$ would be reached much later causing an increase in the waiting time.
Thus, the proportional coefficient should increase together with the support $(b_3 - a_3)$:

$$\varphi_4 = 1 + \delta$$

where $\delta$ indicates the distance $(\, b_3 - a_3\,)$ normalized over the maximum distance possible.

- If $b_3 > a_3$, then the $\delta \in (\,0, 1\,]$ and so $\varphi_4 \in [\,1, 2\,]$ leading to an increase in human waiting time
- If $b_3 < a_3$, then $\delta \in [\,-1, 0\,)$ and so $\varphi_4 \in [\,0, 1\,]$ leading to a decrease in human waiting time

Overall, the formula of Eq. (5.2) provides a coefficient that quantifies the human waiting time in a specific node, allowing to compare many nodes when necessary: the ones with higher coefficient would be targeted as less convenient from a human-wait point of view and would heavily influence the optimization of the cost function (as later shown in Section 5.3.2).

Resuming the parameters of nodes, *Arrival Time* is the time taken to reach the specific node of the reachability tree. If the node is generated by a controllable transition, it has instantaneous firing time and so inherits the arrival time of the father node. If, instead, the node is generated by an uncontrollable transition, its arrival time is computed by the algorithm in two distinct phases, presented in the following.

### 5.2.3 Computation of arrival time

**Phase 1**:
When exploring the reachability tree, node by node the algorithm finds the enabled transitions of such nodes and creates their children. If the transitions generating such children are uncontrollable, the algorithm follows a specific procedure to set their arrival time.
Be:

- $P_{i-1}$: node being explored, with uncontrollable transitions enabled
- $P_i$: one of the children nodes of $P_{i-1}$, generated by uncontrollable transition $t_i$

The way the arrival time of $P_i$ is set depends on two possible cases:

1. transition $t_i$ got enabled in node $P_{i-1}$, i.e. got enabled in the state of the system that is currently being explored. Then, the arrival time in $P_i$ is:
$$t_{Arrival_{P_i}} = t_{Arrival_{P_{i-1}}} + t_{Firing_{t_i}} \tag{5.3}$$
2. transition $t_i$ got enabled in an ancestor node, which is before the node $P_{i-1}$. In this case, it must be taken into consideration the time elapsed between the time instant in which the transition got enabled and the the one in which the transition $t_i$ fired. In this way, it is possible to evaluate if node $P_i$ is really reachable: with this procedure it might be found out that the time related to $P_i$ is incompatible with the system.

For the sake of clarity, an example is given to explain the second point.
In the FTPN that models the system, every uncontrollable transition is represented as a fuzzy number, which is its firing time. As a consequence, this holds also for the reachability tree: every transition is associated to a possibility distribution that models the firing time through a fuzzy number.

Assume that a triangular fuzzy number $[a, b, c]$ is associated to the transition $t_i$ and that such transition got enabled in a node, whose marking is $M_i$.

Thus, the transition $t_i$ will remain enabled for the next nodes (with marking $M_{i+1}, \ldots, M_{i+k}$) if the following rules are fulfilled:

- $a$ $(\,0s \leq a\,)$ is the minimal time that transition $t_i$ must remain continuously enabled, until it can fire.
- $c$ $(\,c \leq \infty\,)$ is the maximum time that transition $t_i$ can remain continuously enabled without firing.

If transition $t_i$ got enabled at time $\tau$ and remains continuously enabled at $\tau + a$, then it can fire.
After time $\tau + a$, transition $t_i$ can remain continuously enabled without firing until $\tau + c$, in which it must be fired or after that time it can no longer fire.
According to the above stated, three scenarios can be pointed out:

1. *actual node $P_{i-1}$ has enabled transitions that potentially bring to nodes with arrival times that are not intersected and that are compatible*

   Suppose that transition $t_i$, with firing time $[\,16s, 17s, 20s\,]$, got enabled in a state having marking $M_i$ and null arrival time. The fact that $t_i$ is enabled does not guarantee that it is effectively going to fire: suppose that another transition $t_j$ is enabled too and got enabled at state $M_i$ as well, with firing time $[\,12s, 13s, 15s\,]$.
   Since $t_j$ must fire within 15 s, it is sure that $t_j$ fires before $t_i$ (note that the firing times are fuzzy - so not deterministic - but the evolution of the system is deterministic, i.e. only one transition at a time can fire).
   When $t_j$ fires, if it does not disable $t_i$ (if it does not remove the tokens that $t_i$ needs to fire) the system is brought into a new state, in which $t_i$ might still be able to fire, if the time taken to reach such state does not exceed the maximum enabling time of $t_i$. In other words, $t_i$ can still fire if the arrival time of $M_{i+1}$ is not greater than 20 s.
   Supposing that the firing of $t_j$ does not disable $t_i$, marking $M_{i+1}$ is reached at least at time 12 s, which is the minimal time for which $t_j$ has remained enabled before firing. At the same time, marking $M_{i+1}$ has been reached at most at time 15 s: hence, when transition $t_i$ fires, a new marking $M_{i+2}$ is reached, at least at time 16 s and at most at 20 s:

   $$t_{Arrival_{M_{i+2}}} = t_{Arrival_{M_i}}\,(\,= 0s\,) + t_{Firing_{t_i}}\,(\,\geq 16s,\, \leq 20s\,)$$

2. *actual node $P_{i-1}$ has enabled transitions that potentially bring to nodes with intersected arrival times and that are compatible with the evolution of the system*

   Suppose to have a transition $t_i$ with firing time $[\,10s, 11s, 20s\,]$ enabled since marking $M_i$, that is the marking of a node with null arrival time.
   Suppose that another transition $t_j$, enabled since marking $M_i$ too, has firing time $[\,12s, 13s, 22s\,]$ and that it fires instead of $t_i$, leading the system to a state with marking $M_{i+1}$. Moreover, suppose again that $t_j$ does not disable $t_i$: in marking $M_{i+1}$, $t_i$ is still enabled.
   For the previous reasoning, the system reaches $M_{i+1}$ at least at 12 s and at most at 22 s; suppose that it is reached at 14 s < t < 15 s. Then, $t_i$ is still enabled but its firing time is not comprehended between 10 s and 20 s anymore, but between 14 s and 20 s: its remaining firing time is less than its initial firing time.
   In fact, $t_i$ got enabled in $M_i$ with firing time $[\,10s, 11s, 20s\,]$ but since the system reached $M_{i+1}$ at least at 14 s, then the time interval $[\,10s, 14s\,]$ is no more a possible firing time for $t_i$.
   This implies that $M_{i+2}$ is reached, through $t_i$, in a time modelled by a fuzzy number comprehended between 14 s and 20 s.
   Such resulting distribution is derived through a method later presented in “phase 2”, that “cuts” the part of arrival time that is not compatible with the evolution of the system.

3. *actual node $P_{i-1}$ has enabled transitions that potentially bring to nodes with intersected arrival times and that are not compatible with the evolution of the system*

   Considering the system of case 2, suppose now that the system reaches $M_{i+1}$ at $21s < t < 22s$ and that the firing of $t_j$ does not take away the resources of $t_i$. Then, it seems that $t_i$ can still fire but, analyzing the time, $M_{i+1}$ is reached at a time instant that exceeds the last possible value of the firing time $t_i$: transition $t_i$ cannot fire, so the node that could have potentially been generated by it, is in reality not reachable. Such node is in fact not added to the reachability tree.

During *phase 1*, the algorithm has to distinguish between the above mentioned scenarios to then assign the arrival time to the interested node, accordingly.

To do so, for each node $P_i$ that it analyzes, it goes through its ancestors to find the one that enabled the transition that generated such node. When the "enabling node" of $P_i$ is found, it is compared with the "father node" of $P_i$:

i. If father node and enabling node coincide, the arrival time of node $P_i$ is set as in Eq. (5.3)

ii. If father node and enabling node do not coincide, it is required to check whether the considered node is reachable. If it is, its time can be set and the node can be added to the tree.
   Be $A = [\,a_1, a_2, a_3\,]$ the possibility distribution of the arrival time of $P_i$ and $B = [\,b_1, b_2, b_3\,]$ the one of its father $P_{i-1}$; it can be stated that for $P_i$ to be reachable, its arrival time must be greater or at least has to be intersected with the arrival time of its father $P_{i-1}$.
   In other words, it must hold:

   $$a_1 \geq b_1$$

   In fact, allowing $a_1$ to precede $b_1$ would mean that it is possible to reach the node $P_i$ before reaching its father node $P_{i-1}$, which is unreasonable. If this happens, $P_i$ is an unreachable node.

   So, considering case ii, first of all the time $A$ that the system would take to reach node $P_i$ has to be computed with Eq. (5.3).
   Then:

   - If the distribution $A$ is not intersected with the distribution of its father, $B$, and its completely after it on the time axis, then $P_i$ is a reachable node and its firing time is not modified.

   - If $A$ is not intersected with $B$ and is completely before it on the time axis, then the system would reach the child before the father: $P_i$ is not reachable.

   - If $A$ is intersected with $B$, the algorithm computes which is the possibility to effectively reach $P_i$ at time $A$ without getting into not plausible situations, since father $P_{i-1}$ was reached at time $B$.

     The possibility that the arrival time of $P_i$ is $A$, is the result of the intersection of two events:

- Arrival time of $P_i$ is $A$
- $P_{i-1}$ is reached before $P_i$, i.e. $B$ is less than $A$

Expressing such concept in a formula:

$$\frac{Poss(\,B < A\,) \cap Poss(\,A\,)}{\max\{\,Poss(\,B < A\,),\, Poss(\,A\,)\,\}} = t_{Arrival_{P_i}} \tag{5.4}$$

where the operation of intersection is performed with the operator "min" (see Chapter 4); the denominator of Eq. (5.4) is a normalizing term, so that the resulting distribution has maximum value in 1.

The possibility to reach $P_{i-1}$ before $P_i$, is expressed as:

$$Poss(\,B < A\,) \quad = \quad \frac{\displaystyle\int_{-\infty}^{x} \mu_A(t)\,dt}{\displaystyle\int_{-\infty}^{+\infty} \mu_A(t)\,dt} \tag{5.5}$$

where $x \in \mathbb{R}$.
The result of Eq. (5.5) is a polynomial curve of the second order, which has to be intersected with the arrival time of $P_i$, $A$.
The result of Eq. (5.4) is a mixture of first and second order curves, with maximum value 1: from this composition of curves, an approximated triangle is derived.
Such triangle would be the possibility distribution of the arrival time of node $P_i$, which is therefore overwritten on its initial distribution, $A$.

The concepts explained up to now are related to *phase 1* of the evaluation of arrival times.
In reality, when evaluating the arrival time at one node it must be checked if the node can be added to the reachability tree not only in terms of plausible time, but also depending on the type of transition that generated it.
In fact, if a node has enabled transitions that are both controllable and uncontrollable, since the controllable ones are prioritized over the uncontrollables, only the children generated by controllable transitions are really added to the reachability tree.
Therefore, the first pruning of the tree consists of excluding the uncontrollable transitions whenever also controllable ones are enabled.

Then, if there are no controllable transitions and there are uncontrollable transitions enabled, the associated nodes must be considered and their time must be computed as in *phase 1*.

If the nodes generated by uncontrollable transitions are more than 1, there might be *conflicts* to establish who is going to fire first. For this evaluation, another method is considered and is referred to as "phase 2": it will eventually overwrite some arrival times or it will lead to declaring some nodes as unreachable.

*Example*:
Suppose that in marking $M_1$ two uncontrollable transitions are enabled: $t_i$ generates node $P_i$ and $t_j$ generates $P_j$.
To analyze the possibility to reach $P_i$ instead of $P_j$, and vice versa, first the arrival time of these two nodes has to be calculated as in *phase 1*. Denoting the arrival time of $P_i$ as $A$ and the one of $P_j$ as $B$, the possible situations are:

1. $A$ completely precedes $B$ on the temporal axis: the conflict is won by the transition leading to $P_i$ and so the arrival time of $P_i$ does not vary, whilst node $P_j$ loses and so is not added to the reachability tree. The possibility to win the conflict for $t_i$ is unitary.

2. Dual case: $A$ completely after $B$, i.e. $A$ loses whilst $B$ is added to the tree with the same time computed in *phase 1* and unitary possibility to win the conflict.

3. $A$ and $B$ are intersected: both $t_i$ and $t_j$ have a chance to win the conflict, each of them with a possibility less than 1 and with different arrival time, calculated as presented in the following.

**Phase 2**:
Given $n$ nodes involved in a conflict, the first step of phase 2 is to compute the arrival time of each node as if it was the winner of the conflict.
At every iteration, the transition $t_i$ generating node $P_i$, $i = 1, \ldots, n$, is assumed to be the winner: the arrival time is computed "given that the considered node won the conflict".
Expressing it in other words, $P_i$ can be the winner of the conflict only if the arrival time of $P_j$, $\forall j = 1, \ldots, n$ and $j \neq i$, is greater than the arrival time of $P_i$.
Therefore, evaluating the arrival time of $P_i$ consists of evaluating how possible is that the arrival time of the other nodes is greater than the one of $P_i$.

Suppose, for example, to have 3 nodes involved in a conflict; be $A$ the distribution of the arrival time of $P_i$, $B$ and $C$ the ones of nodes $P_j$ and $P_k$.
Therefore, the possibility distribution of $P_i$ is given by the intersection of three events:

- Node $P_i$ is reached at a time modelled by $A$
- The other nodes, $P_j$ and $P_k$, are reached at times greater than $A$

Overall, the time distribution of $P_i$ is computed as:

$$Poss(\,C > A\,) \cap Poss(\,B > A\,) \cap Poss(\,A\,) = t_{Arrival_{P_i}} \tag{5.6}$$

where:

$$\begin{aligned} Poss(\,B > A\,) &= \frac{\displaystyle\int_{x}^{+\infty} \mu_B(t)\,dt}{\displaystyle\int_{-\infty}^{+\infty} \mu_B(t)\,dt} \\ Poss(\,C > A\,) &= \frac{\displaystyle\int_{x}^{+\infty} \mu_C(t)\,dt}{\displaystyle\int_{-\infty}^{+\infty} \mu_C(t)\,dt} \end{aligned} \tag{5.7}$$

The curves resulting from Eq. (5.7) are of the second order and are intersected with the triangle $A$, as shown in Eq. (5.6).
The obtained result is then normalized, dividing by the maximum value of the intersection. In this way, it is a distribution with values ranging between 0 and 1; it will be then approximated to a triangle which will eventually be overwritten on the arrival time of $P_i$.

In Appendix it is presented how to solve a conflict with a probabilistic approach, in order to show that the computation required in that case is heavier than the one for possibilistic approach.

Note that, whenever there is a conflict between nodes with intersected arrival times, not only the arrival time must be overwritten, but is also necessary to compute the possibility of winning the conflict.

### 5.2.4 Computation of the arrival possibility

Every node that is added to the reachability tree is characterized by an attribute identifying the possibility to reach the node itself, since some

uncontrollable conflicts are present in the path leading from the root to that node.
Such a possibility is a positive real number, ranging from 0 to 1.
First of all, the possibility to reach a state $M_i$ depends on the possibility to reach its father node $M_{i-1}$; then, if $M_i$ is involved in an uncontrollable conflict, its possibility depends also on the possibility $S$ to win the conflict.

In general, the possibility to reach $M_i$ when it is involved in an uncontrollable conflict is:

$$Poss\,(\,M_i\,) \quad = \quad S \cdot \ \mathrm{Poss}\,(\,M_{i-1}\,) \tag{5.8}$$

where the possibility to win the conflict $S$ expresses how the arrival time of the considered node changed after winning the conflict. In practice, is the ratio between the area under the curve found with Eq. (5.6), before normalizing and approximating it to a triangle, and the area under the curve representing the arrival time distribution determined in *phase 1*.

$$S = \frac{\chi}{\rho} \tag{5.9}$$

where $\chi$ indicates the area under the curve of Eq. (5.6) and $\rho$ indicates the area under the original arrival time distribution.
The formula of Eq. (5.8) can be generally applied to every node: if a node is not involved in an uncontrollable conflict, its "possibility to win the conflict", S, is 1 and so it simply inherits the possibility of its father node to reach the state.

## 5.3 The scheduling algorithm

This Section is going to present how the decision-making problem is handled by the scheduling algorithm implemented.
The cost function is defined and therefore the mechanism of choice of the best actions to command to the robot is shown.
All the steps that follow are subsequent to the generation of the reachability tree (see Section 5.2): indeed, the scheduler chooses the best actions by selecting the best branch of the reachability tree, in terms of cost function.

### 5.3.1 MPC approach

Once identified the criterion for the algorithm to take a decision, i.e. once that the cost function to be optimized has been determined, the

algorithm is able to recognize which is the evolution of the system that allows the most convenient performance.
The algorithm finds the sequence of transitions that bring to the optimal performance, with the aim of sending to the robot commands that will let it chase the optimal plan.

Since the chosen sequence is made of controllable and uncontrollable transitions, it accounts for uncontrollable events which lead to a not fully predictable evolution of the system.
In fact, uncontrollable events like robot actions with ill-known duration or human actions, can be predicted but, in practice, bring the system to a state that cannot be known a-priori.
After commanding the robot and when it has finished the commanded action, it would be meaningless to let it perform the next action that has been previously scheduled because it might not be suitable for the current configuration of the system.

Instead, it would be better to send only the first command of the best sequence, to then re-schedule starting from the new configuration.
This procedure reflects the MPC approach and allows to have a scheduling based on fresh new data.

In practice, when the algorithm returns the sequence of transitions that lead to the best performance, only the first controllable transitions are extracted and converted into commands for the robot.
The output of the algorithm is made of at most 2 commands, that is the case in which both robot arms are free and can perform an action simultaneously.
In the background, there is a thread linking the algorithm to the robot, which sends information on the state of the agents: only when at least one agent is free, the algorithm is invoked to schedule new commands (Figure 5.9).

## 5.3.2 Cost function

Once the reachability tree has been generated, the scheduling algorithm has to choose the branch of the tree that satisfies the most the assigned specifications.
For sure, a priority of the algorithm is to minimize the waiting time of the agents. Every leaf node of the tree has a parameter (*J_Coefficient*) that accounts for *Robot waiting coefficient* and *Human waiting coefficient*, that

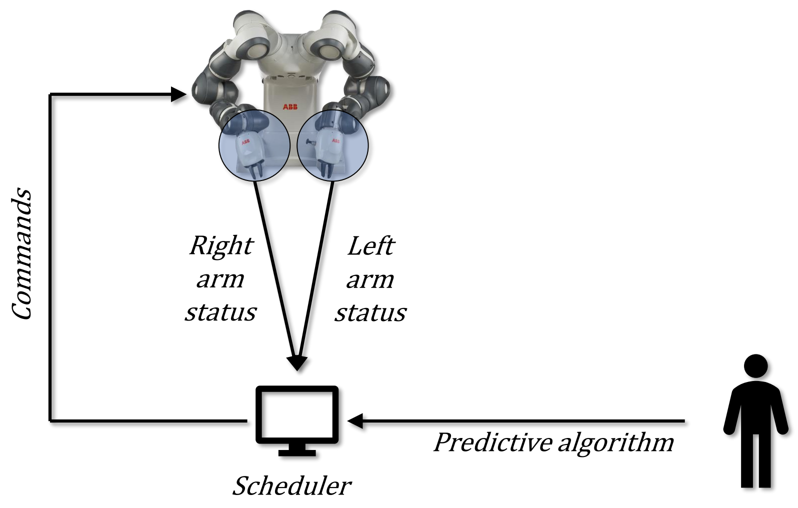


**Figure 5.9:** PC-robot communication

therefore identifies how convenient would be to choose the related branch, from the waiting time point of view.

When evaluating the waiting time of each node, it must be taken into account that every node has its own possibility to be reached: what might seem to be very convenient from a waiting time point of view, might not be very likely to be reached (uncontrollable evolution of the system).

Referring to Figure 5.10, the algorithm might have to choose whether it is more convenient to output $t_{c_1}$ or $t_{c_2}$, assuming that one disables the other.

Choosing $t_{c_1}$, the robot will end up in one of the places generated by $t_{u_1}$, $t_{u_2}$ or $t_{u_3}$, that are 4, 5 or 6. Each of them is characterized by their possibility to be reached: for example, if $t_{c_1}$ is commanded, node 4 is reached with a possibility of 0.8 and, if this happens, the waiting coefficient would be 0.5 (exemplifying value, that is small if the waiting time is small).

It can be noticed that choosing $t_{c_2}$, the waiting time might be much smaller (nodes 8 and 9) with respect to the waiting time resulting from $t_{c_1}$. On the other side, the nodes that seem to optimize the cost function, are characterized by a very low *Possibility to reach the node*: therefore, choosing $t_{c_2}$ there is a possibility to reach nodes with very low waiting time, but it is also (highly) possible to reach a node, 7, with not so convenient waiting time.
Eventually, $t_{c_2}$ might guarantee good performances but the possibility to effectively obtain them is low, i.e. the risk is high.

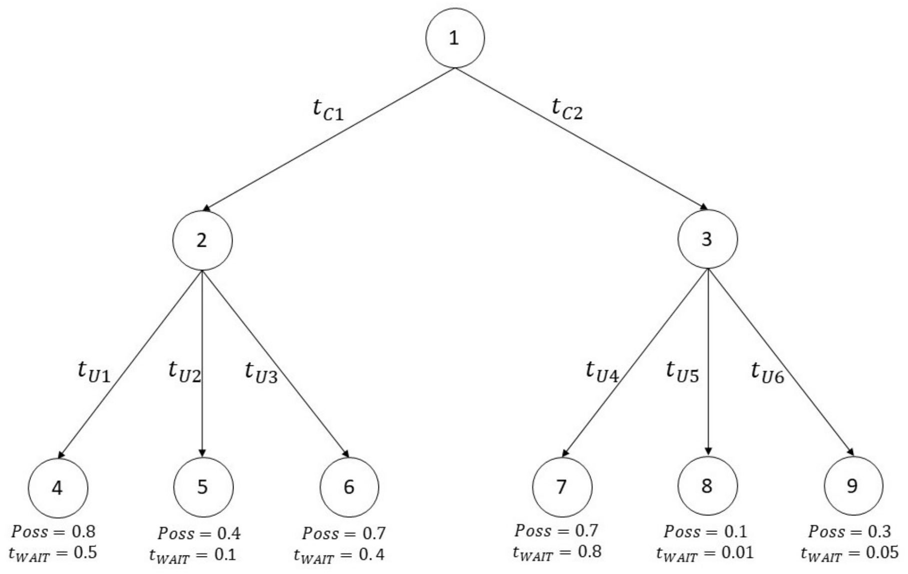

1
$t_{C1}$
$t_{C2}$
2
3
$t_{U1}$
$t_{U2}$
$t_{U3}$
$t_{U4}$
$t_{U5}$
$t_{U6}$
4
5
6
7
8
9
$Poss = 0.8$ $t_{WAIT} = 0.5$
$Poss = 0.4$ $t_{WAIT} = 0.1$
$Poss = 0.7$ $t_{WAIT} = 0.4$
$Poss = 0.7$ $t_{WAIT} = 0.8$
$Poss = 0.1$ $t_{WAIT} = 0.01$
$Poss = 0.3$ $t_{WAIT} = 0.05$

**Figure 5.10:** Evaluation of best branch depending on user's approach to risk

If instead the choice is $t_{c_1}$, the nodes then reached (4,5,6) have not very low waiting times but higher possibilities and so it is more likely that the system evolves as predicted. Having higher possibilities means to include less risk in the choice.

Overall, every user has different criteria to choose, according to his/her needs. For this reason, the cost function is thought in ordered to be "tuned" by the user, that can weigh his/her priorities: the tunable parameter is the *risk coefficient*, which is inversely proportional to the possibility to reach the node.

The risk coefficient is denoted as $\alpha$ and is a real number ranging from 0 to 1.
A low value of $\alpha$ selects a less risky solution of the scheduler, even though it might sacrifice the minimization of waiting times. The higher the value of $\alpha$, the more the choice minimizes the waiting time of the agents.

### 5.3.3 Choice of the best branch

To decide which is the best branch, it is required to determine a parameter that allows to compare all the branches.
Whilst the reachability tree is generated with a breadth-first approach, so starting from the root node towards the leaf nodes, the choice of the

best branch is handled with the opposite procedure: starting from leaf nodes, they acquire a coefficient that expresses the amount of waiting time accumulated up to that node (*J_Coefficient*) and which is propagated upwards until the root node is met.
Considering Figure 5.10, after assigning *J_Coefficient* (in the following it will be explained how) to all the leaf nodes, they are propagated to their father nodes 2 and 3, that will have a *J_Coefficient* based on the *J_Coefficients* and possibilities of their children (because 2 and 3 generate an uncontrollable conflict) and the risk coefficient $\alpha$.
Going to the next level, to the father of 2 and 3, it acquires another *J_Coefficient*: in this case the node generates a controllable conflict, so the propagation of *J_Coefficient* does not depend on the possibility of its children but its the outcome of a choice: node 1 "inherits" the coefficient that maximizes the performances. The transition leading to the node with maximum *J_Coefficient* is the output of the algorithm, and so the command to be sent to the robot.

The same procedure applies for a more complex and general case, with a larger tree, as represented in Figure 5.11, where the red arrows identify controllable conflicts, whilst the black ones identify uncontrollable conflicts.

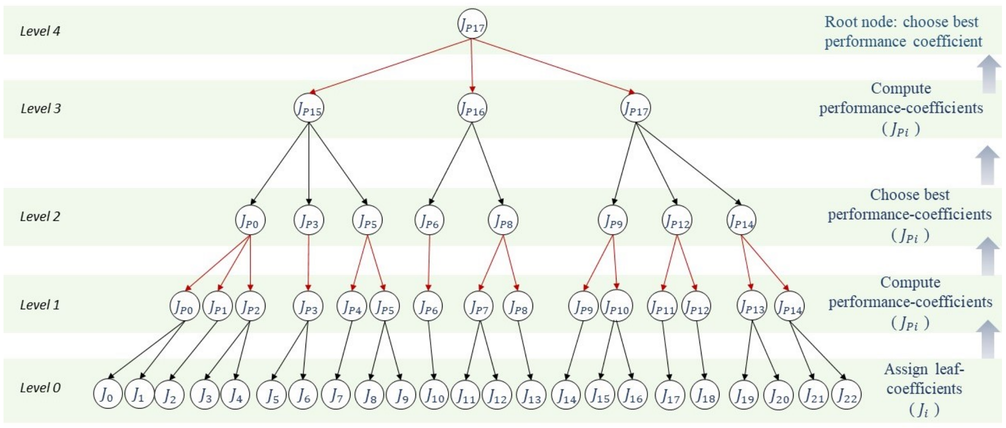

Level 4
Level 3
Level 2
Level 1
Level 0
Root node: choose best performance coefficient
Compute performance-coefficients ($J_{Pi}$)
Choose best performance-coefficients ($J_{Pi}$)
Compute performance-coefficients ($J_{Pi}$)
Assign leaf-coefficients ($J_i$)
$J_{P17}$
$J_{P15}$
$J_{P16}$
$J_{P17}$
$J_{P0}$ $J_{P3}$ $J_{P5}$ $J_{P6}$ $J_{P8}$ $J_{P9}$ $J_{P12}$ $J_{P14}$
$J_{P0}$ $J_{P1}$ $J_{P2}$ $J_{P3}$ $J_{P4}$ $J_{P5}$ $J_{P6}$ $J_{P7}$ $J_{P8}$ $J_{P9}$ $J_{P10}$ $J_{P11}$ $J_{P12}$ $J_{P13}$ $J_{P14}$
$J_0$ $J_1$ $J_2$ $J_3$ $J_4$ $J_5$ $J_6$ $J_7$ $J_8$ $J_9$ $J_{10}$ $J_{11}$ $J_{12}$ $J_{13}$ $J_{14}$ $J_{15}$ $J_{16}$ $J_{17}$ $J_{18}$ $J_{19}$ $J_{20}$ $J_{21}$ $J_{22}$

**Figure 5.11:** Working principle of scheduling algorithm

- At level 0, the *J_Coefficient* for all the leaves is determined:

$$J = \frac{1}{W_H \cdot T_H + W_R \cdot T_R} \tag{5.10}$$

  where $T_H$ and $T_R$ are *Robot_waiting_coefficient* and *Human_waiting_coefficient* respectively and $W_H$, $W_R$ are tunable parameters used to weigh desirably the waiting time of the agents.

- Moving to level 1, the nodes are fathers of uncontrollable conflicts and so their coefficient is determined considering *J_Coefficient* and possibility of their children:

  $$J_{P_i} = \alpha \cdot \max\left( J_k \right) + (1-\alpha) \cdot \sum_{k=0}^{n} J_k \cdot p_k \tag{5.11}$$

  where $J_{P_i}$ is the *J_Coefficient* of a generic node at level 1 or at any other level in which the nodes have outgoing uncontrollable transitions. The node with coefficient $J_{P_i}$ has $n$ children, that have coefficients $J_k$ and possibilities $p_k$, with $k = 0, \dots, n$.
  The coefficient $\alpha$ stands for the risk desired by the user: if $\alpha = 1$, the risk is maximum and therefore the user prefers to consider only the best case of the evolution of the system, without considering the related possibilities. If $\alpha = 0$ the user prefers to find a compromise between optimization of waiting time and significant values of possibility.

- At level 2, the nodes generate controllable transitions, so they have to choose the best performing coefficient:

  $$J_{P_i} = \max\left( J_k \right) \tag{5.12}$$

  where $J_{P_i}$ is the coefficient of the generic node at level 2 (or at any other level in which nodes generate controllable transitions); $J_k$ is the *J_Coefficient* of the $k-th$ child of $J_{P_i}$, $k = 0, \dots, n$.

Referring to Figure 5.11, the leaf nodes with coefficient $J_0$ and $J_1$ have the same father, with coefficient $J_{P_0}$ and are involved in an uncontrollable conflict.
Therefore, assuming that:

- $J_0$ has *Human_ Waiting_Coefficient* $= 0.3$,
  *Robot_ Waiting_Coefficient* $= 0.1$,
  *Possibility_to_reach_the_node* $= 0.4$

- $J_1$ has *Human_ Waiting_Coefficient* $= 0.2$,
  *Robot_ Waiting_Coefficient* $= 0.1$,
  *Possibility_to_reach_the_node* $= 0.8$

the respective coefficients are:
$J_0 = \dfrac{1}{0.3+0.1} = 2.5 \qquad J_1 = \dfrac{1}{0.2+0.1} = 3.33$

If $\alpha = 0$, $J_{P_0} = \sum_{k=0}^{n} J_k \cdot p_k = J_0 \cdot p_0 + J_1 \cdot p_1 = 3.33 \cdot 0.8 + 2.5 \cdot 0.4 = 3.64$.
If $\alpha = 1$, $J_{P_0} = \max(J_k) = 3.33$.
If $\alpha = 0.6$, $J_{P_0} = 0.6 \cdot \max(J_k) + (1 - 0.6) \cdot (J_0 \cdot p_0 + J_1 \cdot p_1) = 0.6 \cdot 3.33 + 0.4 \cdot 3.64 = 1.998 + 1.456 = 3.454$.

Assume to perform the same computation to obtain $J_{P_1}$ and $J_{P_2}$, with $\alpha = 0.6$, and to obtain $J_{P_1} = 3.27$ and $J_{P_2} = 2.98$. Then, the coefficient of the father node at level 2 is the maximum between $J_{P_0}$, $J_{P_1}$ and $J_{P_2}$ and so $J_{P_0}$.

The pseudo-code reported in Algorithm 1 and Algorithm 2 summarizes the steps involved in the computation of a new scheduled plan.
The notation adopted for presenting the pseudo-code is described in the following:

**Root_Node**: node that represents the initial state of the system, with marking $M_0$
**PRE** and **POST**: incidence matrices of the FTPN
**DATA**: information coming from predictive algorithm and from the communication thread (see Section 5.3.1) with the robot
**Unprocessed_nodes**: set of nodes that have been generated and still have to be analyzed
**This_Node**: node that the algorithm is currently analyzing
**Children_list**: set of children nodes of the considered one
**Horizon**: temporal scheduling horizon over which the tree ceases to expand
**set_time_phase1 ()**: method to set arrival time to the considered node, according to Eq (5.4)
**set_time_phase2 ()**: method to set arrival time to the considered node, according to Eq (5.6)
**set_possibility_after_conflict ()**: method to set the possibility to reach the considered node, basing on Eq. (5.8)
**Choose_Best_Path ()**: method that computes the best path (5.3.3)
**Update_data ()**: method that updates the data related to the FTPN, computing the new marking reached by the system - according to the above-mentioned data received - and overwriting the firing time of transitions that express the duration of actions. In fact, whenever an action is performed, its duration is measured and stored, in order to have a more precise estimate of its duration
**Create_Net ()**: method that creates the FTPN
**Get_Optimal_Plan ()**: method that creates the reachability tree from

the created FTPN
**List_Of_New_Samples**: information updated at each iteration of the algorithm
**Find_Best_Branch ()**: method that selects the most performing branch of the tree
**List_Best_Sequence**: list of best controllable transitions taken from the output of Find_Best_Branch (); they correspond to the commands to send to the robot

**Algorithm 1:** Get_Optimal_Plan ()

```
create Root_Node;
add Root_Node to Unprocessed_nodes;
while Unprocessed_nodes has elements do
    This_node = first element of Unprocessed_nodes;
    if This_node is not generated by Human Action transition then
        if This_node.Arrival_time < Horizon then
            if This_node has enabled transitions then
                if enabled transitions are uncontrollable then
                    for every enabled transition do
                        create node generated by the transition;
                        add node to Children_list of This_Node;
                    end
                    if This_node has children then
                        for every child do
                            child.Arrival_time = set_time_phase1 ();
                            if time computed at phase 1 is plausible then
                                if child is involved in an uncontrollable conflict then
                                    child.Arrival_time = set_time_phase2 ();
                                    child.Possibility_to_reach_the_node =
                                      set_possibility_after_conflict ();
                                else
                                    child.Possibility_to_reach_the_node = inherit
                                      possibility from father;
                                end
                            else
                                delete child;
                            end
                        end
                    end
                else
                    // enabled transitions are controllable
                    for every controllable transition do
                        create node generated by the transition;
                        add node to Children_list of This_Node;
                        node.Possibility_to_reach_the_node = inherit possibility from father;
                    end
                end
                if This_node does not have children then
                    delete This_node;
                else
                    for every child do
                        if This_node can not generate then
                            Compute_Robot_waiting_coefficient ();
                            Compute_Human_waiting_coefficient ();
                        end
                        add child to Unprocessed_nodes;
                    end
                end
            else
                delete This_node from Unprocessed_nodes;
            end
        else
            delete This_node from Unprocessed_nodes;
        end
    else
        delete This_node from Unprocessed_nodes;
    end
end
List_Best_Sequence = Choose_Best_Path ();
return List_Best_Sequence;
```

**Algorithm 2:** Choose_Best_Path ()

```
1  Compute the list of leaf nodes;
2  for every leaf node do
3      assign J__Coefficient = $\frac{1}{WT_H + WT_R}$ ;
4  end
5  for every leaf node do
6      initialize current__node to leaf node;
7      while current__node != Root__Node do
8          if current__node is generated by a controllable transition then
9              father = select father of current__node;
               // set J_ Coefficient as in Eq.5.12
10             father.J__Coefficient = best__J;
11         else
               // set J_Coefficient as in Eq.5.11
12             father.J__Coefficient = $(\alpha) \cdot best\_J + (1 - \alpha) \cdot Sum$;
13         end
           // Iterative procedure
14         current__node = father;
15     end
16 end
17 Best__Node = child of Root__Node with maximum J__Coefficient;
18 Best__Sequence = Best__Node.Find__Best__Branch ();
19 return Best__Sequence;
```

# Chapter 6

# Experimental results

This Chapter is aimed at reporting the results obtained by testing the developed scheduling algorithm.
In Section 6.1 the goal of the experiments is presented; Section 6.2 describes the experimental setup, introducing in detail the working areas and the objects used, together with an explanation of the operations that form the assembly processes. Moreover, the FTPN used to model the system is illustrated and described.
Section 6.3 specifies the conditions under which the experiments have been performed and the related outcomes.
In particular, the performance of the proposed scheduling algorithm is compared with the ones obtained using a less accurate scheduler, which is presented in the same Section.
In Section 6.4, other results obtained from simulation tests are provided, showing the efficiency of the proposed algorithm in terms of computational costs; moreover, it is aimed at proving its efficiency also in adapting to different conditions of the system and different tuning of parameters.

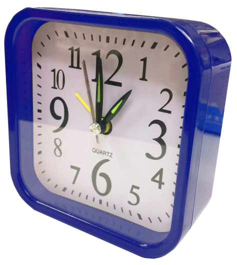
## 6.1 Goal of the experiments

In order to test the effectiveness of the scheduler, in this Chapter a realistic use case of human-robot collaborative assembly process has been considered.
An industrial robot co-operates with a human operator in an assembly line, to produce objects of common use: some of the actions required to obtain the final products are performed by the human and the others by the robot.
Since the actions of an agent depend on the success of the actions of the

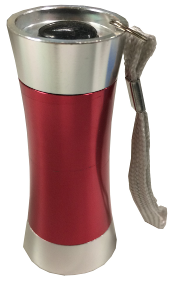
other agent, the overall process is intended as a collaboration.

For the considered experiments, the products selected are of different kind and made of many components, in order to simulate and test a complex and uncertain process; indeed, the assembly operations are copious and subject to unpredictable variations in their duration, due to the collaborative context.
As a consequence, the aforementioned test case has a great amount of uncertainty, which allows to verify the efficiency of the proposed algorithm, that has to control the robot guaranteeing an optimal synchronization of the actions of all agents involved in the cooperation.
In particular, the robot has to promptly satisfy the requests made by the human operator, carrying out the tasks that allow him/her to fluently work without wasteful waits.

The products chosen for the assembly line are torches and clocks, shown in Figure 6.1 and whose components are listed in Table 6.1; in the following, details about the assembly processes will be shown.



(a)



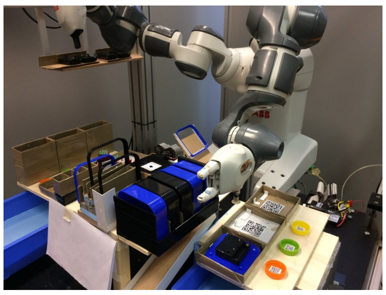
(b)

**Figure 6.1:** Assembled products

## 6.2 Use case description

### 6.2.1 Experimental setup

The collaborative robot used in the experiments is the dual-arm ABB YuMi: it is equipped with a suction tool and a parallel gripper, that are used to pick and place the components of the objects to be assembled.
The two arms of YuMi are considered as two independent entities, which can work simultaneously: the only constraint to impose about the motion of YuMi, is the non-collision of the arms. Therefore, if a collision is predicted to happen, one of the two arms stops to allow the other to continue its action.
For this reason, a specific area called "Assembling Shared Area" has been defined: it is a region that can be accessed by both arms (Figure 6.7). The arm that intends to enter the region, first has to require its availability; if the region is occupied by the other arm, the one sending the request has to wait until it becomes free. As a consequence, if an arm is performing an action and at some point requests to enter the Assembling Shared Area, which is instead occupied, then the time required to complete the current action increases considerably, according to the time for which the other arm occupies the area.
In turn, the presence of the shared area leads to an increase in the uncertainty affecting the duration of robot actions.
A practical example of the Assembling Shared Area being occupied by an arm is given in Figure 6.17, for the right arm, and Figure 6.2, for the left arm.


(a)

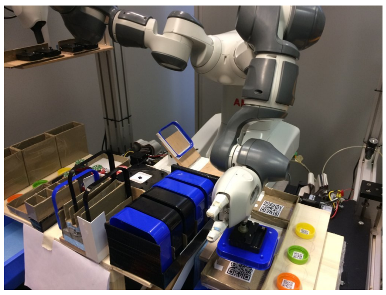
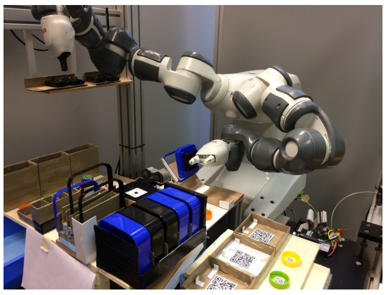

(b)

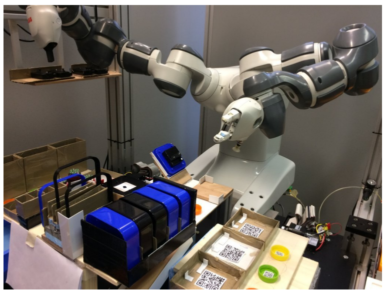
**Figure 6.2:** Action L3 ( part 1 )

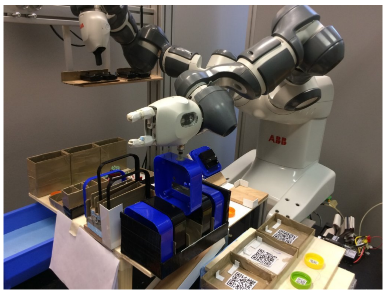
*Figure 6.2: continued on next page.*

*Figure 6.2: continued from previous page.*

**(c)**


**(d)**


**(e)**


**(f)**

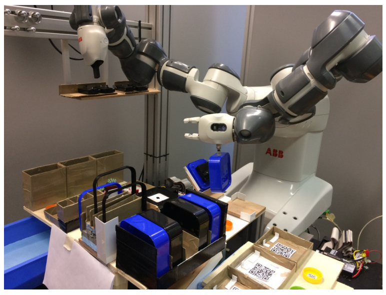
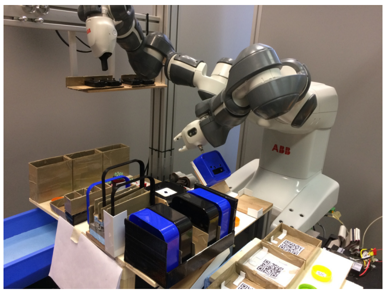

**(g)**

**Figure 6.2:** Action L3 ( part 2 )

*Figure 6.2: continued on next page.*

*Figure 6.2: continued from previous page.*


**(h)**

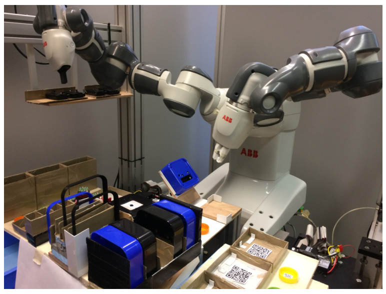

**(i)**

**Figure 6.2:** Action L3 ( part 3 )

The end effectors of YuMi are embedded with two cameras, used to detect specific objects in the working areas.
This is used in pick-and-place operations, to know where the object that has to be picked is located, or to find a free spot to place an object.

A Microsoft Kinect depth camera is used to acquire the positions of the human's hands, to capture the moment in which he/she is starting or performing an action. Such information is sent to the scheduler, which updates the state of the underlying FTPN.
At the same time, this information is required to know if the human is entering some areas that are common regions, used by both human and robot. In fact, when a human deposits a piece or takes one from an area that can be used by the robot too, the latter slows down its speed to avoid collision with the operator and - at the same time - to allow him/her to serenely conduct his/her tasks, without fearing the closeness of robot arms.

In the physical setup of the test case, the common regions, i.e. areas in which the robot is entitled to slow down, are:

- *Central Common Area, Common Left Area*: they are intermediate stations in which the human places the pieces that he/she assembled, and that will be picked by the robot to continue the assembly process. These regions can be entered by both agents and therefore are part of those regions in which the camera has to recognize if the human is inside them, so that the robot is commanded to slow down its motion.
- *Ramp Area, Common Right Area*: they are common areas, in which

the robot deposits the pieces that has assembled and that have to be picked by the human operator. As the above areas, these are programmed to be controlled by the camera, to let the robot slow down in case of simultaneous operations inside them.

Other components of the setup are:

- *BF_complete_torch*: buffer containing assembled torches.
- *BF_complete_clock*: buffer containing assembled clocks.
- *Robot Working Station (RWS)*: station in which the robot temporarily deposits a piece to perform an assembly operation on it. In the following, the stations $RWS_{R_1}, \dots, RWS_{R_N}$ will identify the working stations used by the robot when performing actions $R_1, \dots, R_N$ respectively, and $RWS_{L_1}, \dots, RWS_{L_N}$ will correspond to actions $L_1, \dots, L_N$.
- *Buffer (BF)*: to perform any action, pieces are needed and therefore they are all collected in buffers, one for every specific component. As an example of notation, $BF_{1_{R_1}}, \dots, BF_{N_{R_1}}$ will denote the $N$ buffers used by the robot to perform action $R_1$; $BF_{1_{H_1}}, \dots, BF_{N_{H_1}}$ will be the buffers for the human action $H_1$, etc.

### 6.2.2 Assembly operations

In this Section, a detailed description of the operations carried out by the agents of the system will be given, showing the procedure that each of them follows.
The considered use case satisfies the hypotheses made in Chapter 3: it comprehends Robot Actions, Human Actions, Robot to Human Actions and Human to Robot Actions.
The actions performed by the right arm are reported as "Action R", those of the left arm are "Action L"; the actions of the human are denoted as "Action H".
The logical order of the actions to assembly a torch and a clock is explained in the following and represented at the end of the Chapter:

1. *Action R1* (Robot to Human Action): the right arm of the robot takes a ring from $BF_{1_{R_1}}$ with the gripper and deposits it on the working station $RWS_{R_1}$. Then, picks a glass from $BF_{2_{R_1}}$ with the vacuum tool and puts it inside the ring. With the gripper again picks a light from $BF_{3_{R_1}}$ and puts it in the piece assembled up to now, to

then push it towards the human operator, letting it slide on a ramp that has been built specifically for this purpose, which is the Ramp Area.
The piece assembled at this stage is identified as “piece 1”.
The sequence of operations is represented in Figure 6.17.

2. *Action H1* (Human to Robot Action): the human takes “piece 1” from the Ramp Area, then takes the body of the torch from $BF1_{H_1}$ and assembles them, forming “piece 2”, which will be placed in Common Left Area (see Figure 6.18).

3. *Action H2* (Human to Robot Action): the human picks three batteries from $BF1_{H_2}$ and the cage from $BF2_{H_2}$, to insert the first ones in the latter. The assembled piece (“piece 3”) is put in Central Common Area (see Figure 6.19).

4. *Action L1* (Robot Action): the robot takes “piece 2” from Common Left Area and places it in $RWS_{L_1}$ (see Figure 6.20).

5. *Action R2* (Robot to Human Action): takes “piece 3” from the Central Common Area and puts it inside “piece 2”, that now is in $RWS_{L_1}$. The completed product (“piece 4”) is moved and placed in Common Right Area (see Figure 6.21).

6. *Action H3* (Human Action): the human takes “piece 4” from Common Right Area, the top of the torch from $BF1_{H_3}$ and assembles them, forming the final product, which is therefore put in BF_complete_torch (see Figure 6.22).

7. *Action R3* (Robot to Human Action): the robot takes the clock face from $BF1_{R_3}$ to place it on the working station $RWS_{R_3}$; then, takes the engine of the clock from $BF2_{R_3}$ and mounts it on the clock face, forming “piece 5” and depositing it in Common Right Area (see Figure 6.23).

8. *Action H4* (Human to Robot Action): the human picks “piece 5” from Common Right Area and screws it. Then, takes the needles from $BF1_{H_4}$ and attaches them to “piece 5”, forming “piece 6”, which is placed in Common Left Area (see Figure 6.24).

9. *Action L2* (Robot Action): the robot clamps the frame of the clock from $BF1_{L_2}$ and places it in $RWS_{L_2}$. Then, uses the vacuum to pick the glass from $BF2_{L_2}$ and assembles everything, forming “piece 7” (see Figure 6.25).

10. *Action L3* ( Robot Action ): the robot grabs "piece 6" from Common Left Area and wedges it in "piece 7", forming "piece 8"; then it uses the vacuum to pick the back of the clock in $BF_{1_{L_3}}$ and to place it on top of previous piece, forming "piece 9" (see Figure 6.2).

11. *Action R4* ( Robot to Human Action ): the robot grabs "piece 9" and places it in Common Right Area (see Figure 6.26).

12. *Action H5* ( Human Action ): the human takes "piece 9" and screws it, to then put it in the final buffer BF_complete_clock (see Figure 6.27).

All the components needed to assemble the above mentioned pieces are shown in Table 6.1. In all the actions in which the robot accesses Common Right Area or Common Left Area, the first operation performed by the robot is to take a picture of the common area from above, in order to read the QR code placed inside it. This operation allows the robot to know if there is at least one piece in the common area or if there is a free place to deposit a piece.
As it is shown in Figure 6.7, the Common Right Area is divided into three areas, defined as A, B and C, each of which is used to contain "piece 4", the "piece 5" and "piece 9", respectively. Similarly, the Common Left Area is divided into two areas, A and B, to contain "piece 2" and "piece 6", respectively.
The experiment performed on the real assembly plant is characterized by a specific sequence of operations. The human performs cyclically the same sequence of actions, without varying the order, so that the predictive algorithm can reach a steady state condition: in this way, the information about the duration of human actions transferred to the scheduling algorithm is more reliable.
Indeed, if the human changes instead the order of actions at each iteration, the predictive algorithm would constantly be in a training phase and would return predictions that are far from reality, i.e. not consistent.

### 6.2.3 Use case modeling

The chosen assembly process is modeled through a FTPN. Referring to the notions exposed in Chapter 3, the physical system and its working principle can be represented as in the FTPN reported at the end of this Chapter in Figure 6.28 and Figure 6.29, whose places and transitions are summarized in Tables 6.3, 6.4 and 6.5.

All the actions are represented in the FTPN as “Action in progress” places (Chapter 3), having an input arc coming from a controllable transition, e.g. $Start_{R_1}$, that identifies the start of an action, and having an output arc, going towards an uncontrollable and timed transition that represents the duration of such action, modeled as a possibility distribution, e.g. $End_{R_1}$ (see Figure 6.3).
In addition, a human action can only start if the human has first expressed the intention to start executing the action; in fact, when the human manifests an intention (e.g. start to assemble the torch), the robot performs those actions that are necessary to provide him/her with the pieces needed to start the intended action.

In summary, a human action can be split in three stages:

- $Intention_{H_X}$: is an intention expressed by the human operator, when his/her aim is to start a specific action. The intention can be manifested even though the pieces necessary to perform the intended action are yet not available;

- $Start_{H_X}$: is the actual start of the intended action. It can happen only if all the resources that have to be used in such action are available for the human operator and only if the place that has to host the final product is free;

- $End_{H_X}$: identifies the fulfillment of the action, carrying the information about the duration of such action, represented as a fuzzy number;

where $X = 1, \dots, 5$.

In Figure 6.3 a representation of actions of the robot and actions of the human is given, to highlight the difference in their structure. Note that Robot Actions and Robot to Human Actions have the same structure; the same holds for Human Actions and Human to Robot Actions.

Referring to Figure 6.3, the “availability of resources” required to satisfy a human intention stands for the availability of pieces in buffers ($BF1_{H_X}, \dots, BF_{N_{H_X}}$) or of pieces provided by Robot to Human Actions. Each human action has different requirements, which are shown in Figure 6.4 and Figure 6.5.

The scheduler is invoked every time that at least one of the two arms is free and returns the best sequence of actions to be sent to the robot.

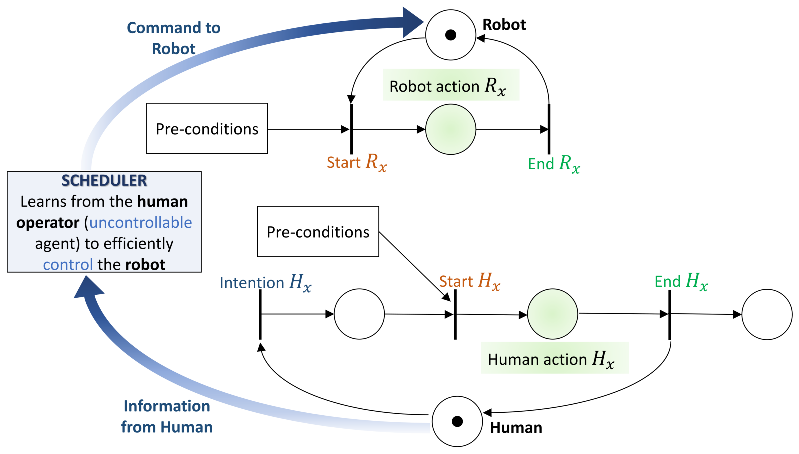


**Figure 6.3:** Structure of human and robot actions

In Figure 6.6, the layout of the experimental setup is shown, specifying all the components and areas mentioned above.

Notice that Common Right Area and Common Left Area, represented in Figure 6.6, are further divided into regions as mentioned in Section 6.2.2: their structure is represented in Figure 6.7.

## 6.3 Presentation of the experimental results

The efficiency of the algorithm developed for this thesis has been proved comparing it with a simpler scheduler, which accounts for uncertainty in a less precise way.
The simpler scheduler considered goes under the name of "uniform scheduler": it represents the duration of actions as intervals, i.e. uniform distributions rather than fuzzy, and does not consider how possible is to effectively reach a node of the generated reachability tree.
Indeed, neglecting the possibility and representing roughly the duration of tasks, the uniform scheduler is expected to be less performing than the fuzzy one.
For the experiments two different assembly processes have been considered and they are identified by the following patterns:

- **Pattern A**: both torches and clocks are assembled. The human operator performs an action to assemble a torch, alternated with another action to assemble a clock; thus, the related intentions are shown by the human operator in the following order:

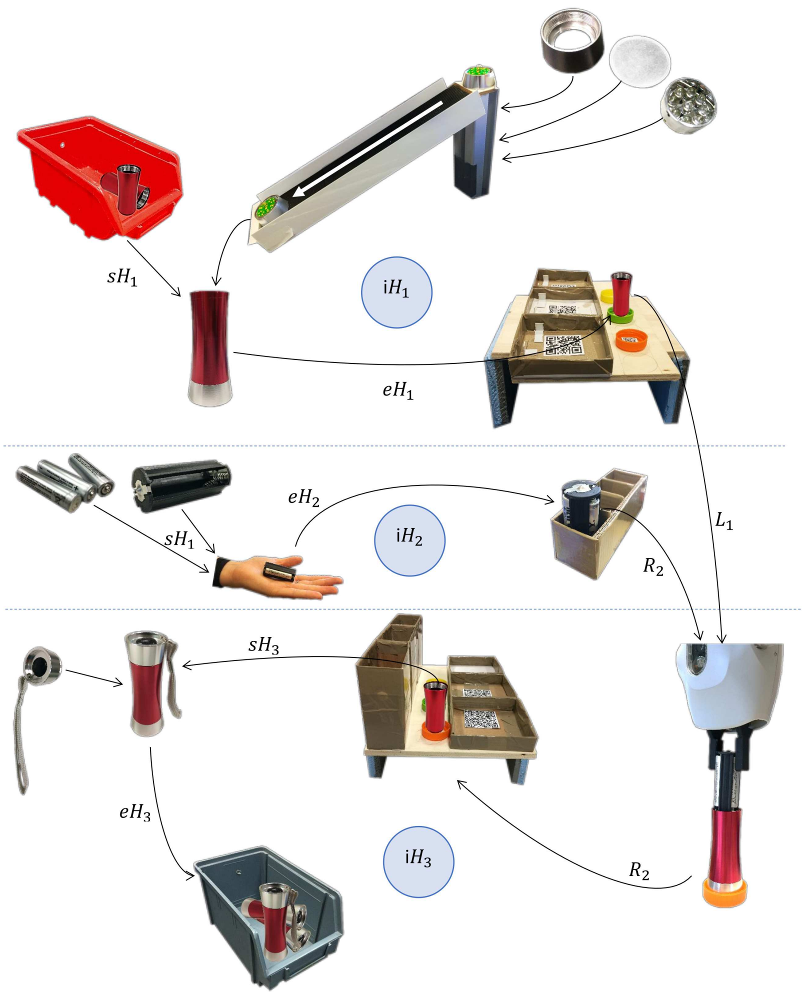


**Figure 6.4:** Resources needed to perform human actions in assembling torches

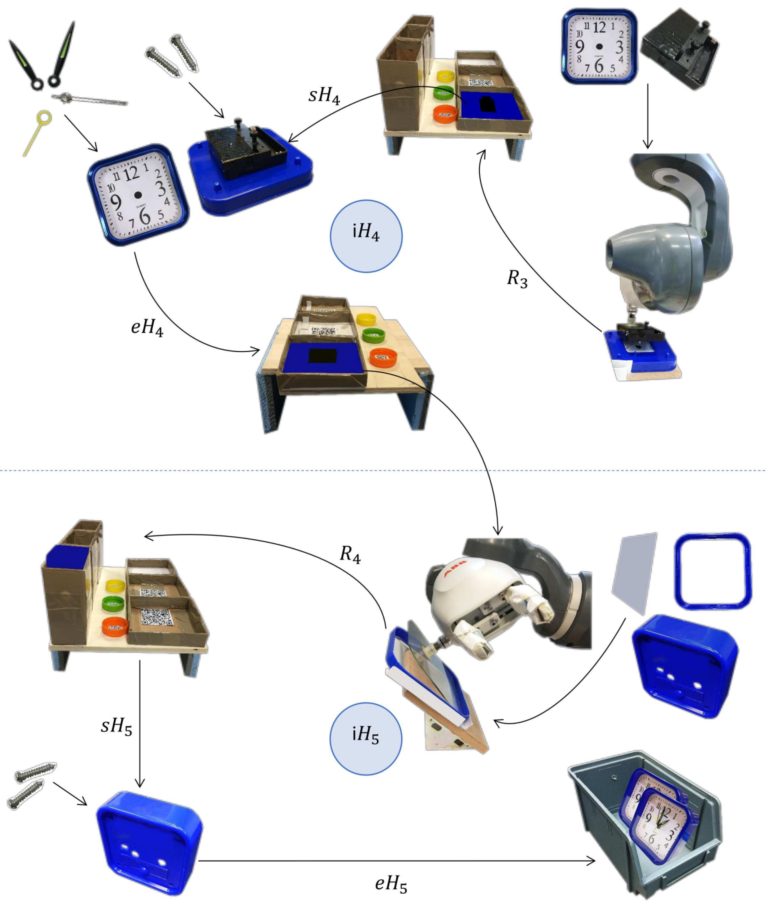


**Figure 6.5:** Resources needed to perform human actions in assembling clocks

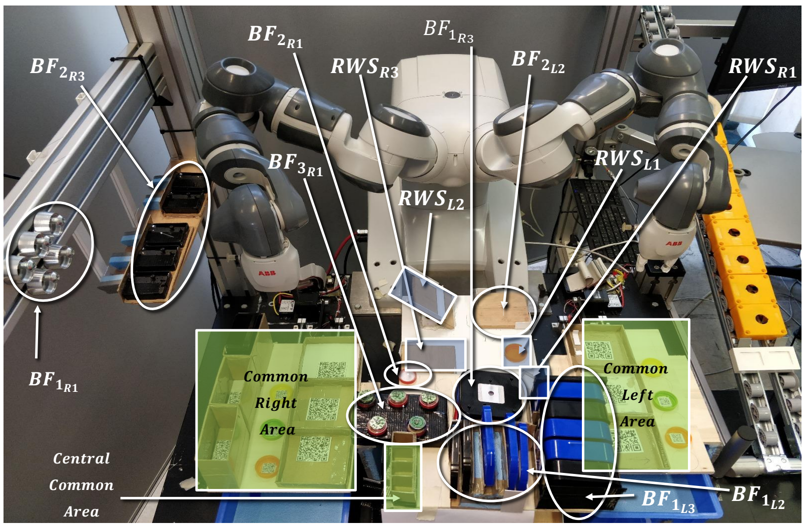


**Figure 6.6:** Layout of the experimental setup

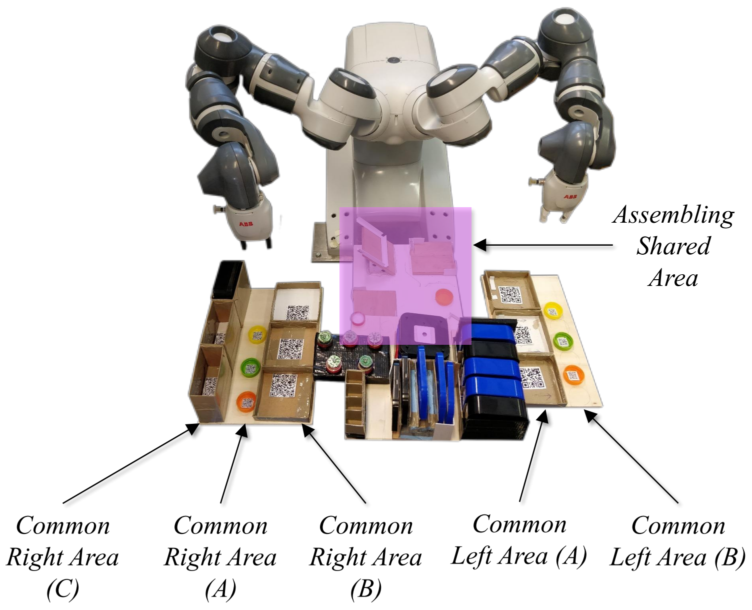


**Figure 6.7:** Regions of the working area

  - $IntentionH_1$
  - $IntentionH_4$
  - $IntentionH_2$
  - $IntentionH_5$
  - $IntentionH_3$

- **Pattern B**: only torches are assembled, hence the order of intentions shown is:
  - $IntentionH_1$
  - $IntentionH_2$
  - $IntentionH_3$

For every pattern, it has been evaluated the total amount of time for which the human operator has been waiting during the production of a single piece, both in fuzzy and uniform approaches.
The time required to produce one piece is computed as the time elapsed between two consequent completed products (*production cycle*). The human waiting time computed from the experiment performed with pattern A is represented in Figure 6.8, together with the robot waiting time; analogously, the waiting times obtained from the experiment with pattern B are represented in Figure 6.9.
On each box plot reported, the red line indicates the median value; the bottom and the top edges of the box identify the 25th and 75th percentiles, respectively; the '+' symbols represent extreme data points, which are outliers.

From above results, it can be noticed that both agents have, in average, reduced waiting time when a fuzzy scheduler is used, rather than a uniform one:

- Pattern A: waiting time decreased of 22.55%
- Pattern B: waiting time decreased of 26.28%

Figure 6.10 shows how the time required to produce one torch and one clock is distributed, both with fuzzy and uniform approach (Figure 6.10).

Considering the same set of operators working when testing both fuzzy and uniform schedulers, the production time of objects in the assembly line is lower when the scheduling algorithm is fuzzy.

By testing, it has been noticed that:

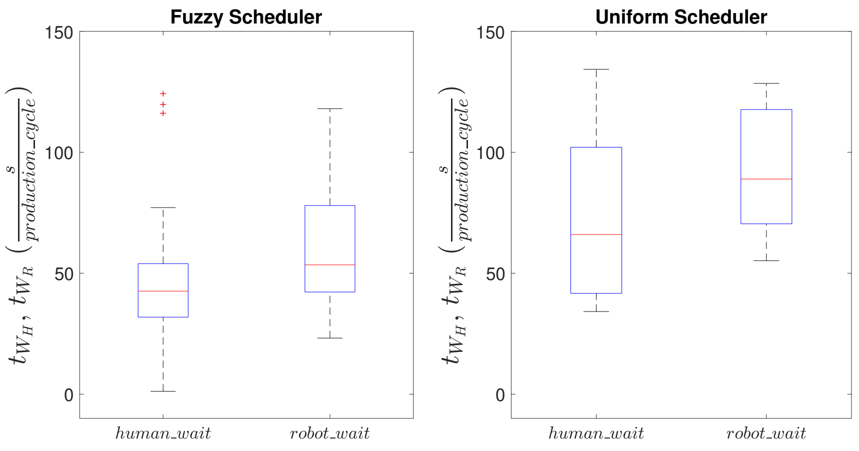


**Figure 6.8:** Waiting time of agents in pattern A: fuzzy and uniform approaches

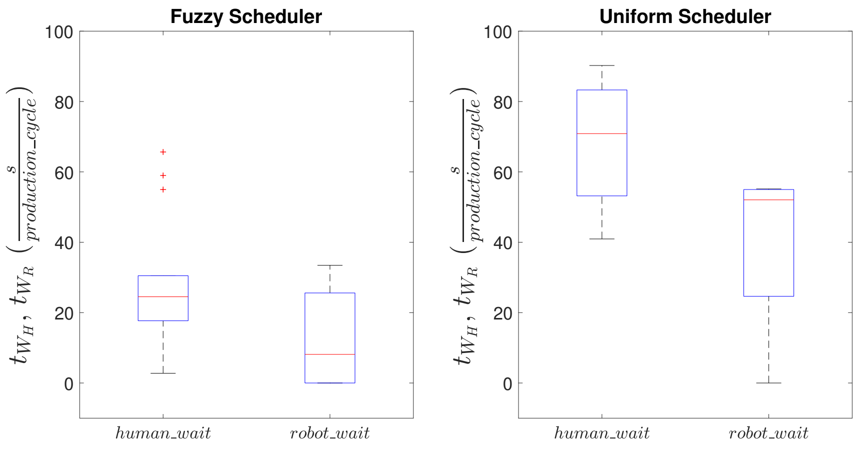


**Figure 6.9:** Waiting time of agents in pattern B: fuzzy and uniform approaches

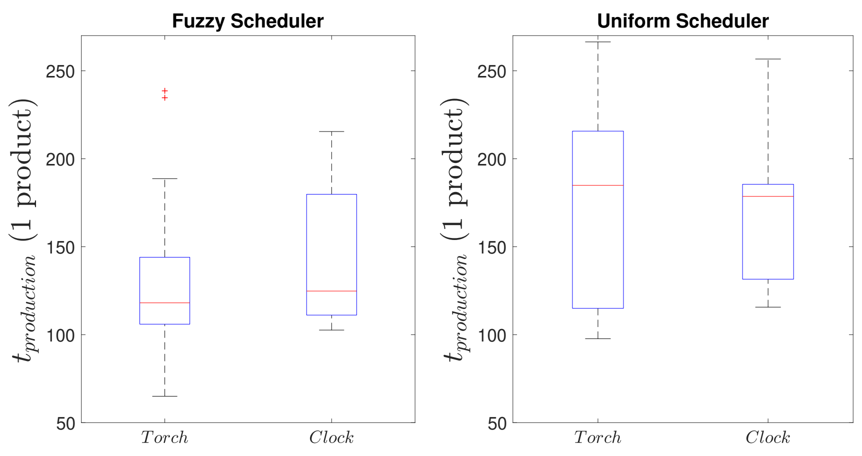


**Figure 6.10:** Production time for one torch and one clock: fuzzy and uniform approaches

- Production time for one torch reduced by 36.12%
- Production time one clock reduced by 30.14%

This shows that the developed algorithm is able to better adapt to the human's needs, having positive consequences on the overall performance. Indeed, since the fuzzy scheduler handles the possibilities ($\alpha = 0$), it avoids to command actions that would ideally lead to a state with minimum waiting time but that, at the same time, would have a very low possibility to be reached.
Thus, cutting out from the choices those branches of the reachability tree that are not reliable (very low possibility), the fuzzy scheduler is less likely to "get wrong"; the uniform scheduler, instead, might command an action believing that it would bring the system in a state with minimum waiting time, without considering that the evolution to such state is almost unfeasible.
As a consequence, it may happen that - with a uniform scheduler - the system evolves to a state different from the expected one, leading to an increase in the waiting time and therefore in the overall production time.

In the following another experimental result is reported, to justify the choice of a fuzzy scheduler.
The duration of the robot actions have been timed during experiments on patterns A and B: in Figure 6.11 and Figure 6.12 the plots on the left show how the samples of the duration of robot actions are distributed.

It can be noticed that the samples are very scattered, proving that the duration of actions is effectively uncertain and unpredictable: from here, the choice not to represent such duration as a uniform distribution, i.e. avoiding to assume that the samples are homogeneously distributed in a time interval.
For every box plot representing the duration of an action, the respective triangular fuzzy number has been represented (according to the method presented in Section 5.1), as illustrated in Figure 6.11 and Figure 6.12, on the right.

## 6.4 Offline results

Offline experiments have been performed in order to test the adaptability of the scheduling algorithm to different conditions: tests were made varying both initial conditions of the system and risk coefficient $\alpha$.

### Case 1: varying the initial conditions

The test reported in the following proves that varying the initial conditions of the system, the algorithm returns the expected command for the robot, i.e. the command that satisfies the preconditions required by the expressed human intention.
Given the initial configuration:

- *Common_Right_Area*, *Common_Left_Area*, *Central_Common_Area*, *Ramp_Area* empty
- RWSs all empty
- Risk coefficient $\alpha = 0$
- Scheduling horizon = 60 s
- Human expresses $IntentionH_1$

in this scenario the human operator expresses the intention to perform $ActionH_1$ and he/she can do it only if “piece 1” (see Section 6.2.2) is available. Since “piece 1” is the outcome of $ActionR_1$, the expected command for the robot provided by the algorithm should be $StartR_1$.
As a result, the algorithm returns commands $StartR_1$ and $StartL_2$, satisfying the request made by the human; to show how such choice is taken by the scheduler, part of the analyzed reachability tree is reported in Figure 6.13. It shows the *J_Coefficients* and possibilities of each node: the nodes of the selected sequence are the ones with the highest performance coefficients (*J_Coefficients*).

The same intention has been tested providing a different configuration of the system: it has been assumed that the human operator has assembled “piece 3” (see Section 6.2.2) which therefore is in Central Common Area; then, there is a “piece 2” in $RWS_{L_1}$.
In summary, the initial configuration is:

- *Common_Right_Area*, *Common_Left_Area*, *Ramp_Area* are empty
- *Central_Common_Area* is occupied by “piece 3”

- $RWS_{L_1}$ is occupied by "piece 2", while other RWSs are empty
- Risk coefficient $\alpha = 0$
- Scheduling horizon = 60 s
- Human expresses $IntentionH_1$

In this scenario the right arm of the robot can perform more actions: as a consequence, the reachability tree developed from the root node and shown in Figure 6.14 has more branches than in the previous case. In this configuration, the right robot arm is entitled to perform $StartR_1$, $StartR_2$ and $StartR_3$.

In other words, in this case the scheduler has to choose between more alternatives, i.e. the algorithm has to analyze a much wider reachability tree, making it more complicated to identify the branch that effectively maximizes the performances.
It has been instead proven that, even in this case, the scheduler is efficient: when the operator shows the will to start $ActionH_1$, the scheduler keeps the focus on his/her intention and satisfies the request.
Indeed, the output of the algorithm is still $StartR_1$ and $StartL_2$, as highlighted in Figure 6.14.

**Case 2: varying risk coefficient**

The test reported in the following concerns tuning the risk coefficient $\alpha$, which can be set by the user: it is proven that different branches are selected by the fuzzy scheduler, according to $\alpha$.
For very low values of $\alpha$ the user favours a less risky and more conservative approach: then, the scheduler outputs commands that find a trade-off between low waiting times and sufficiently high possibilities that the system evolves as predicted (see Section 5.3.3).
For high values of $\alpha$, the scheduler opts for a different plan, that aims at an evolution of the system towards states characterized by minimum waiting time, irrespective of the possibility to effectively reach those states.
In conclusion, as the coefficient varies, the algorithm outputs commands that change from case to case, proving that it is able to adapt to every case and preference.

Given the following configuration of the system:

- *Central_Common_Area*, *Common_Right_Area*, *Common_Left_Area* empty
- RWSs all empty
- BFs full
- Scheduling horizon = 60 s
- Human expresses intention to start $ActionH_1$

first it has been assumed $\alpha = 0$.
In Figure 6.15 part of the reachability tree is represented; in particular, the branch following the best sequence is shown. The output of the algorithm consists of commands $StartR_1$ and $StartL_2$, which in turn satisfies the request made by the human.

Considering a more risky approach, the coefficient has been set as $\alpha = 1$. The output of the algorithm is still $StartR_1$ and $StartL_2$: in Figure 6.16, it is shown that coefficients propagated through the reachability tree up to this choice are different with respect to those of Figure 6.15.
This is due to the fact that the risky approach does not consider the influence of possibilities in computing the *J_Coefficients*.

For instance, in Figure 6.16 it can be noticed that the *J_Coefficient* of *NodeK* is 0.455 which is obtained by choosing the maximum between the *J_Coefficients* of its children nodes $NodeK_1$ and $NodeK_2$:

$$J_K = \max\left( J_{K_1}, J_{K_2} \right)$$

where,

$$J_{K_1} = 0.402, \qquad J_{K_2} = 0.455$$

In Figure 6.15, the *J_Coefficient* of *NodeK* is 0.405 which is obtained as weighted sum of the *J_Coefficients* of its children nodes $NodeK_1$ and $NodeK_2$:

$$J_K = J_{K_1} \cdot Poss_{K_1} + J_{K_2} \cdot Poss_{K2}$$

where,

$$J_{K_1} = 0.402, \qquad Poss_{K_1} = 1$$

$$J_{K_2} = 0.027, \qquad Poss_{K_2} = 0.098$$

In Table 6.2 the scheduling time of the proposed algorithm is illustrated, varying the temporal scheduling horizon.

| | Scheduling time | |
|---|---|---|
| | **Mean** | **Variance** |
| ***Horizon = 100 s*** | 1.8000375 | 0.01446521 |
| ***Horizon = 80 s*** | 0.8902685 | 0.003487785 |
| ***Horizon = 60 s*** | 0.3231295 | 3.8111385e-04 |
| ***Horizon = 50 s*** | 0.20840675 | 3.4786611e-04 |
| ***Horizon = 40 s*** | 0.11516825 | 1.7121111e-04 |
| ***Horizon = 30 s*** | 0.046589375 | 5.8740619e-04 |
| ***Horizon = 20 s*** | 0.036883825 | 2.1913742e-04 |

**Table 6.2:** Scheduling time of the proposed algorithm

As it can be noticed, the scheduling time is very low, leading to a fast algorithm.
Indeed, the fuzzy scheduler is capable of creating a thinner reachability tree: whenever finds branches in which the leaf nodes have too low possibility to be reached, the scheduler prunes the tree. In this way, the least possible branches are removed, leading to a faster generation and analysis of the tree, guaranteeing a lower computational cost.
A uniform scheduler can be interpreted as a specific case of the fuzzy one, when the most risky approach is considered ($\alpha = 1$): it minimizes the waiting coefficients, irrespective of possibilities, turning out to be more imprecise and therefore less efficient.

According to the experiments reported, the results prove that the algorithm implemented in this thesis is able to adapt to many situations and conditions.
Overall, it reveals to be a compelling alternative to traditional scheduling methods, since it provides fast results and good performances.

| Resource | Meaning | Biding place | Meaning |
|---|---|---|---|
| $P_0$ | Human | $P_{50}$ | Human Wait H3 |
| $P_1$ | Left Arm | $P_{48}$ | Human Wait H2 |
| $P_2$ | Right Arm | $P_{46}$ | Human Wait H4 |
| $P_3$ | Common Right Area (A) | $P_{49}$ | Human Wait H5 |
| $P_4$ | $BF_{1_{L2}}$ | $P_{47}$ | Human Wait H1 |
| $P_5$ | Common Left Area (A) | | |
| $P_6$ | Common Left Area (B) | | |
| $P_7$ | Common Right Area (C) | | |
| $P_8$ | Common Right Area (B) | | |
| $P_9$ | $BF_{2_{L2}}$ | | |
| $P_{10}$ | $BF_{1_{L3}}$ | | |
| $P_{11}$ | $BF_{1_{R3}}$ | | |
| $P_{12}$ | $BF_{2_{R3}}$ | | |
| $P_{13}$ | $RWS_{L1}$ | | |
| $P_{15}$ | $RWS_{R3}$ | | |
| $P_{16}$ | $BF_{1_{R1}}$ | | |
| $P_{17}$ | $BF_{2_{R1}}$ | | |
| $P_{18}$ | $BF_{3_{R1}}$ | | |
| $P_{19}$ | $RWS_{R1}$ | | |
| $P_{20}$ | Central Common Area | | |
| $P_{33}$ | $RWS_{L2}$ | | |

**Table 6.3:** Meaning of places of FTPN of Figure 6.28 and Figure 6.29

| Action in progress | Meaning | Other places | Meaning |
|---|---|---|---|
| $P_{23}$ | Action L1 | $P_{21}$ | Piece 3 in Central Common Area |
| $P_{25}$ | Action R2 | $P_{22}$ | Piece 2 in Common Left Area (A) |
| $P_{27}$ | Action R3 | $P_{24}$ | Piece 2 in $RWS_{L1}$ |
| $P_{30}$ | Action L3 | $P_{26}$ | Piece 4 in Common Right Area (A) |
| $P_{32}$ | Action L2 | $P_{28}$ | Piece 5 in Common Right Area (B) |
| $P_{35}$ | Action R4 | $P_{29}$ | Piece 6 in Common Left Area (B) |
| $P_{38}$ | Action R1 | $P_{31}$ | Piece 7 in $RWS_{L2}$ |
| $P_{39}$ | Action H1 | $P_{34}$ | Piece 9 in $RWS_{L2}$ |
| $P_{40}$ | Action H2 | $P_{36}$ | Piece 9 in Common Right Area (C) |
| $P_{41}$ | Action H4 | $P_{37}$ | Piece 1 in Ramp Area |
| $P_{42}$ | Action H5 | | |
| $P_{43}$ | Action H3 | | |
| $P_{44}$ | WaitRightArm | | |
| $P_{45}$ | WaitLeftArm | | |

**Table 6.4:** Meaning of places of FTPN of Figure 6.28 and Figure 6.29

| Controllable | Meaning | Controllable | Meaning |
|---|---|---|---|
| $t_1$ | Start L1 | $\tau_0$ | Start H1 |
| $t_3$ | Start R2 | $\tau_2$ | End L1 |
| $t_8$ | Start R3 | $\tau_4$ | Start H2 |
| $t_{10}$ | Start L3 | $\tau_5$ | End R2 |
| $t_{12}$ | Start L2 | $\tau_6$ | Start H3 |
| $t_{14}$ | Start R4 | $\tau_7$ | Start H4 |
| $t_{18}$ | Start R1 | $\tau_9$ | End R3 |
| $t_{25}$ | Start Wait Right Arm | $\tau_{11}$ | End L2 |
| $t_{26}$ | Start Wait Left Arm | $\tau_{12}$ | End L3 |
| | | $\tau_{15}$ | End R4 |
| | | $\tau_{16}$ | Start H5 |
| | | $\tau_{17}$ | End R1 |
| | | $\tau_{19}$ | End H1 |
| | | $\tau_{20}$ | End H2 |
| | | $\tau_{21}$ | End H4 |
| | | $\tau_{22}$ | End H5 |
| | | $\tau_{23}$ | End H3 |
| | | $\tau_{24}$ | End Wait Right Arm |
| | | $\tau_{27}$ | End Wait Left Arm |
| | | $\tau_{28}$ | Intention H4 |
| | | $\tau_{29}$ | Intention H1 |
| | | $\tau_{30}$ | Intention H2 |
| | | $\tau_{31}$ | Intention H5 |
| | | $\tau_{32}$ | Intention H3 |

**Table 6.5:** Meaning of transitions of FTPN of Figure 6.28 and Figure 6.29

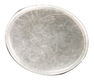
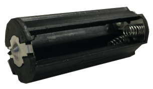
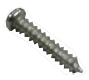
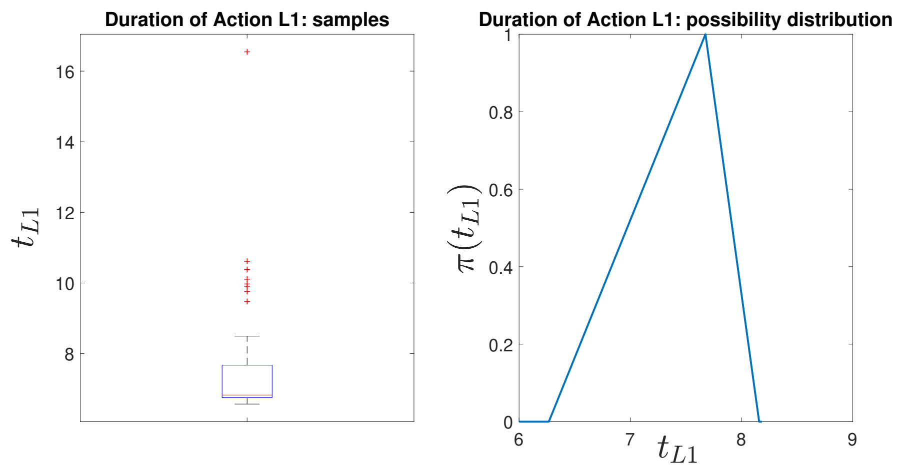
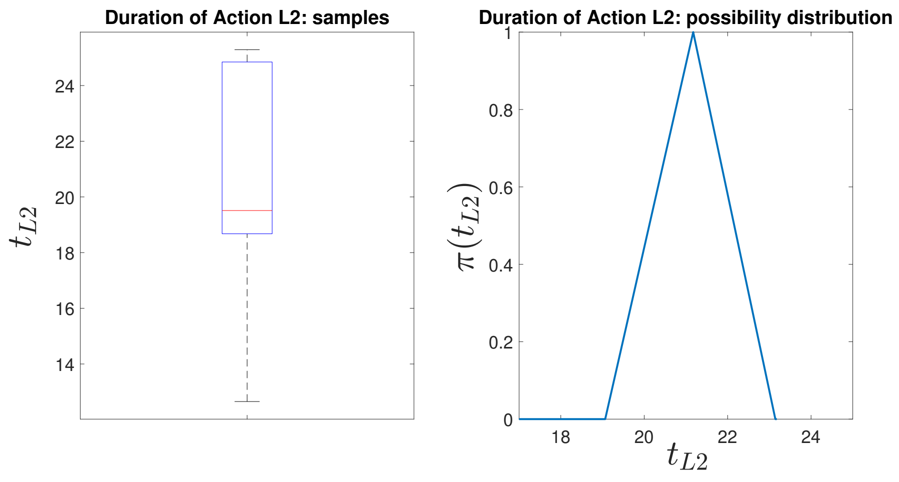
| **Torch Components** | **Clock Components** |
|---|---|
| Ring | Needles |
| Glass  | Clock Face |
| Light | Clock Glass |
| Batteries | Clock Frame |
| pile Cage  | Back |
| Torch body | Engine |
| Top | Screws  |

Table 6.1: Components of torches and clocks



**(a)**

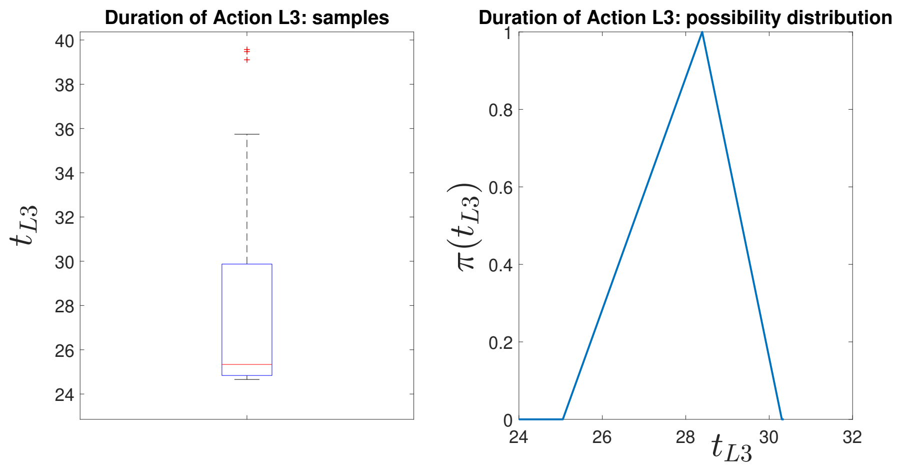


**(b)**

**Figure 6.11:** Duration of left arm actions: scattered samples and fuzzy representation ( part 1 )

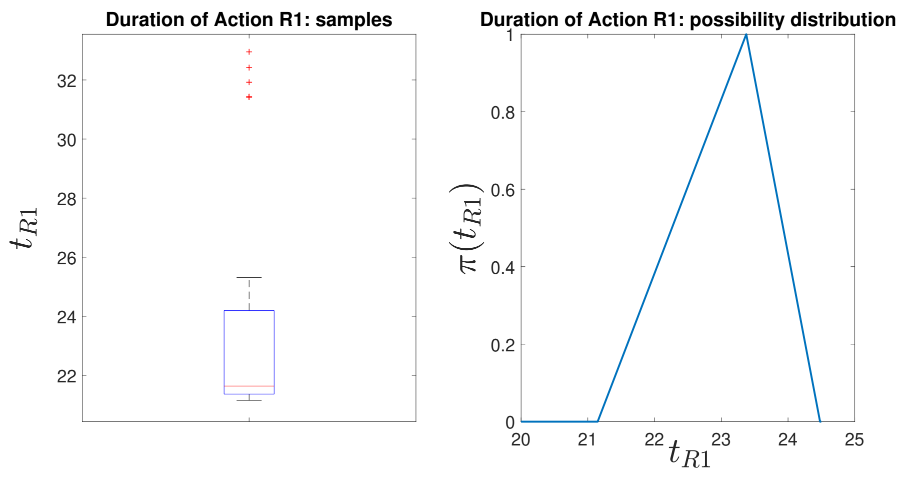
*Figure 6.11: continued on next page.*

*Figure 6.11: continued from previous page.*

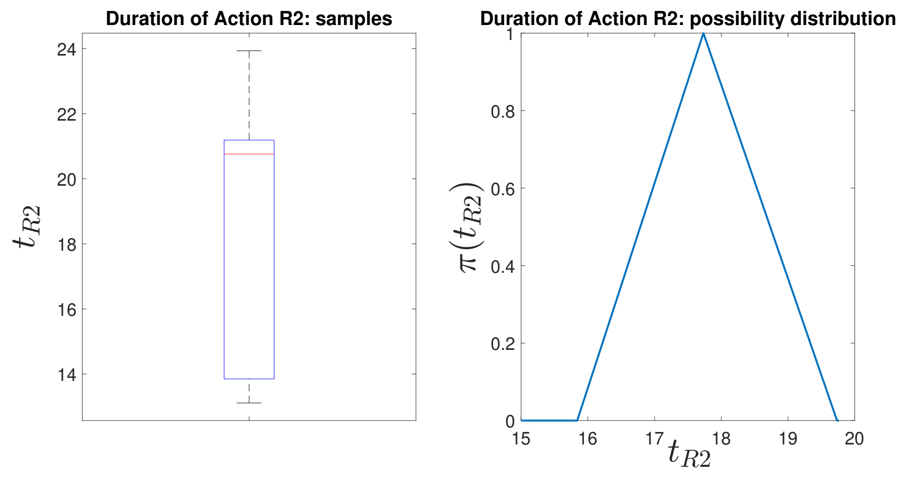


**(c)**

**Figure 6.11:** Duration of left arm actions: scattered samples and fuzzy representation ( part 2 )



(a)

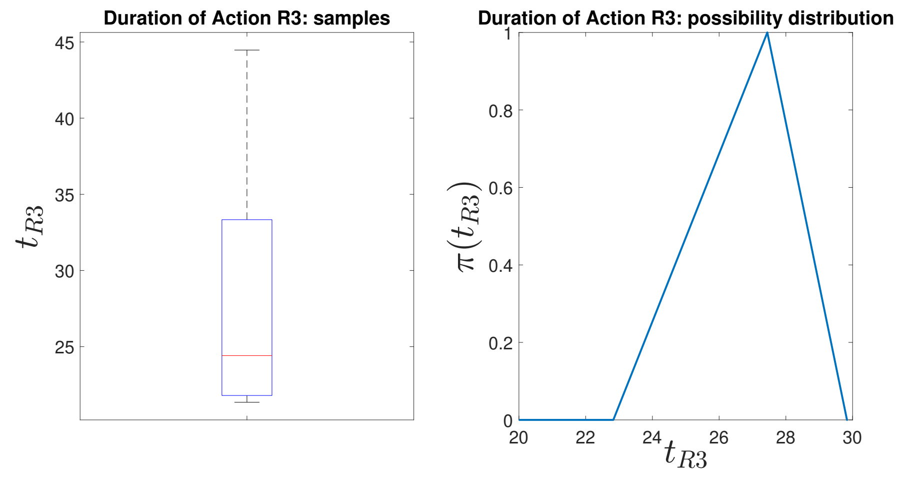


(b)

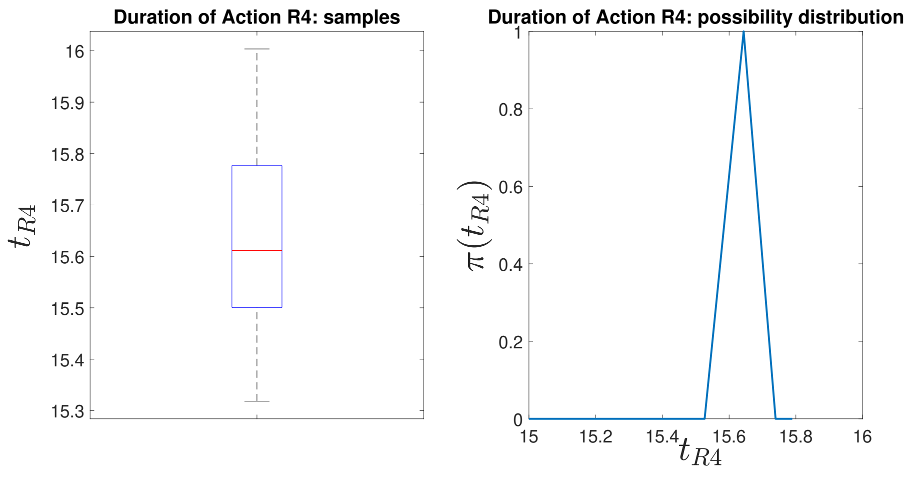
**Figure 6.12:** Duration of right arm actions: scattered samples and fuzzy representation ( part 1 )

*Figure 6.12: continued on next page.*

*Figure 6.12: continued from previous page.*



**(c)**



**(d)**

**Figure 6.12:** Duration of right arm actions: scattered samples and fuzzy representation ( part 2 )

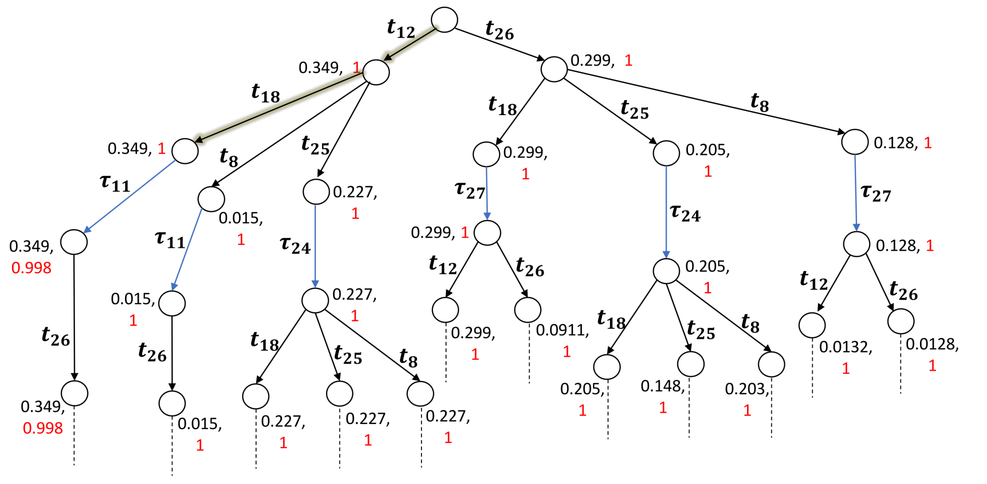


**Figure 6.13:** Branch following the best sequence: initial configuration

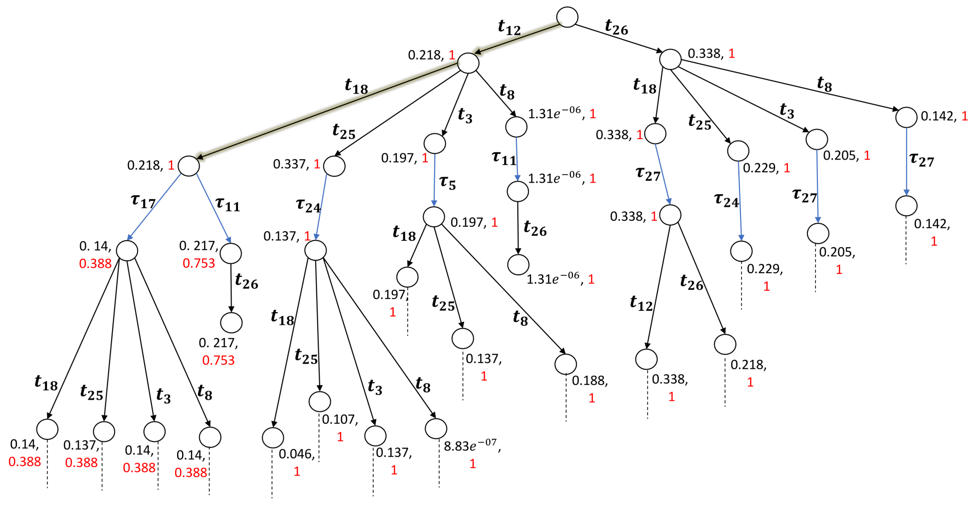


**Figure 6.14:** Branch following the best sequence: alternative configuration

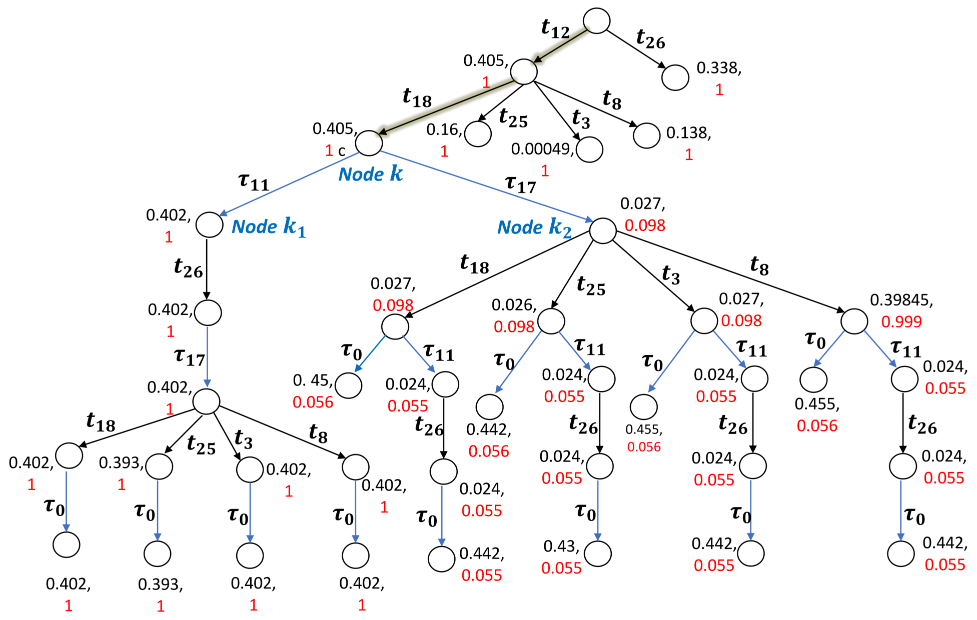


**Figure 6.15:** Branch following the best sequence ($alpha = 0$)

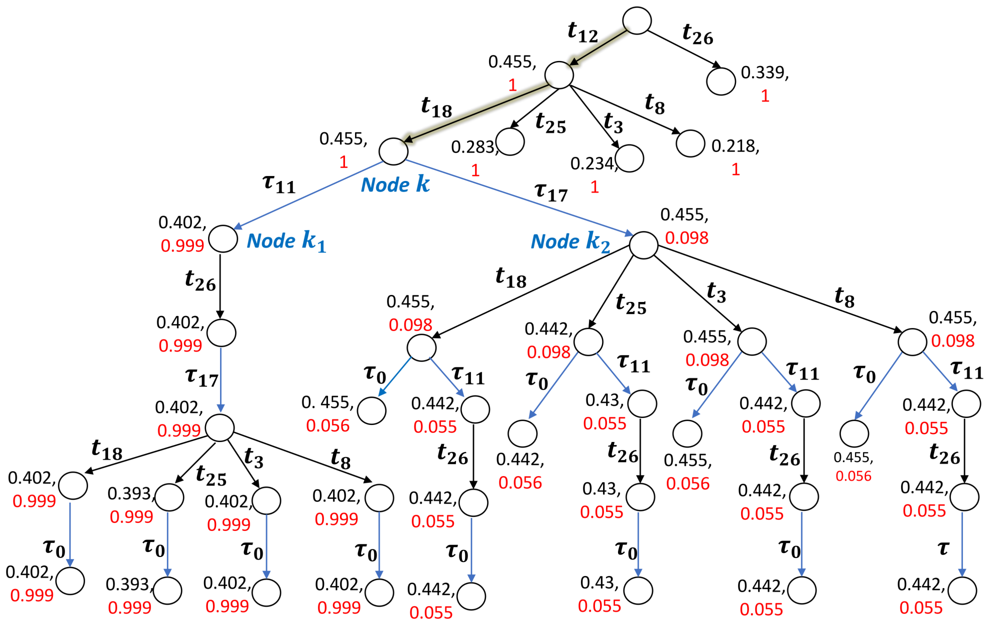
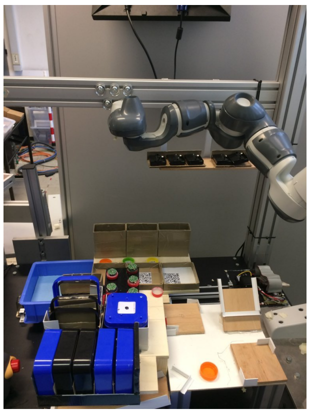
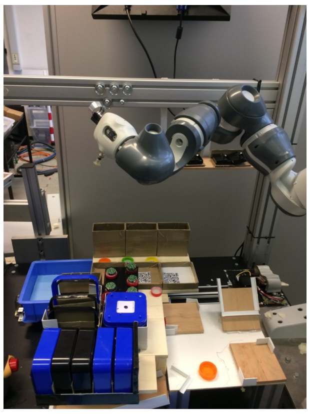


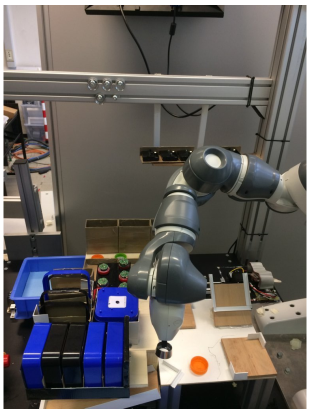
**Figure 6.16:** Branch following the best sequence ( $alpha = 1$ )



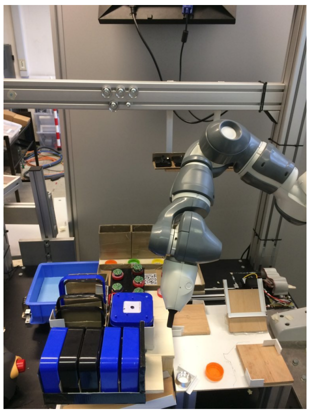
**(a)**



**(b)**



**(c)**

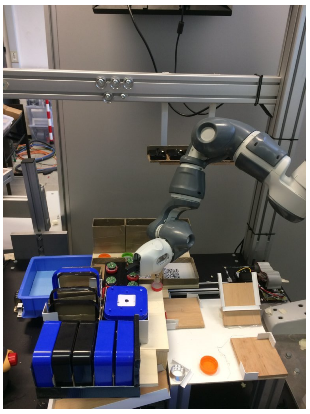


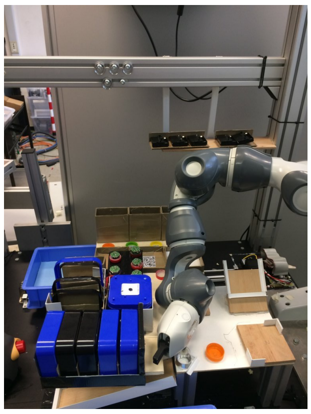
**(d)**

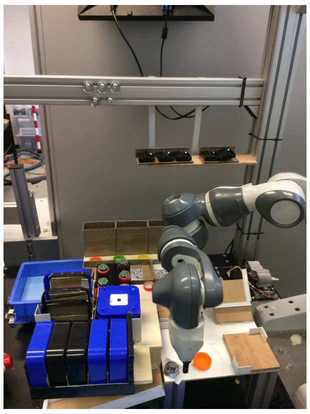
**Figure 6.17:** Action R1 ( part 1 )

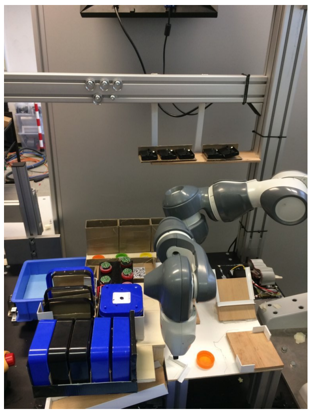
*Figure 6.17: continued on next page.*

*Figure 6.17: continued from previous page.*


**(e)**


**(f)**


**(g)**


**(h)**

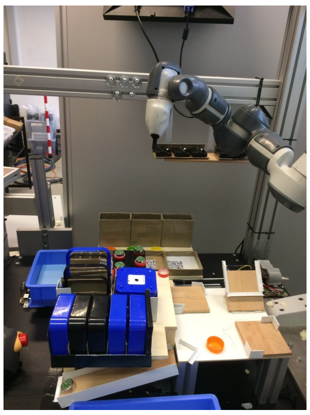
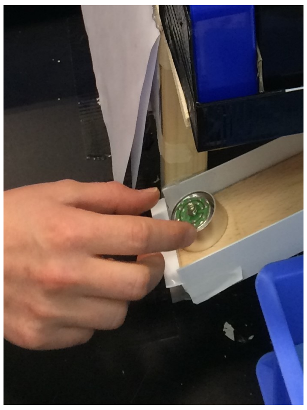
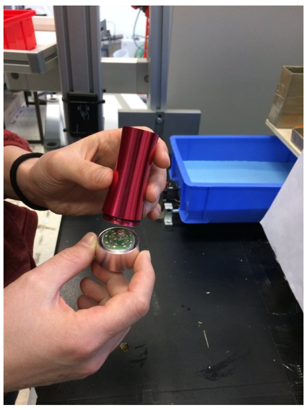

**(i)**

**Figure 6.17:** Action R1 ( part 2 )


(a)


(b)

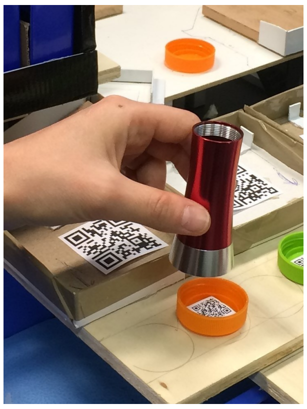
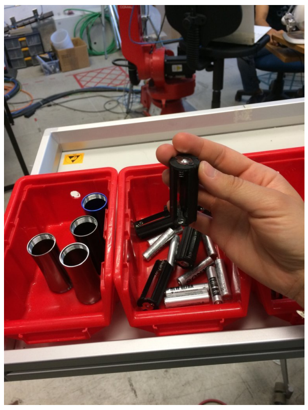
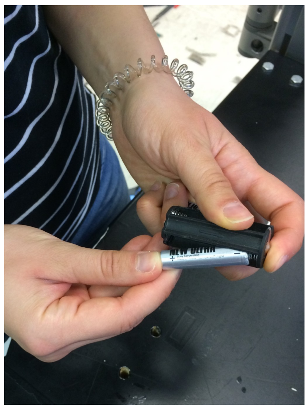
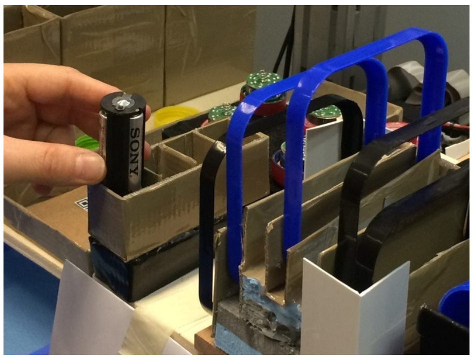

(c)

**Figure 6.18:** Action H1

(a)

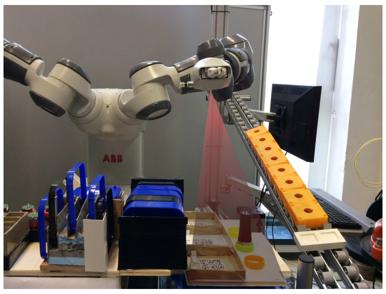

(b)

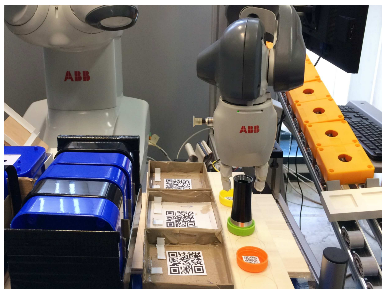

(c)



(d)

**Figure 6.19:** Action H2


(a)


(b)

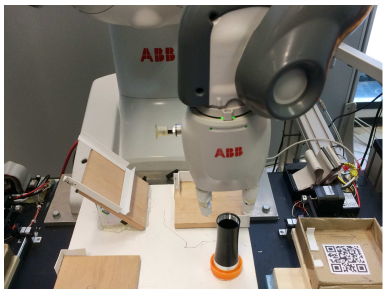
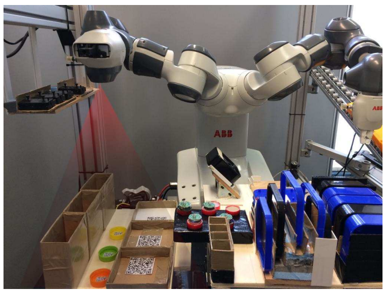
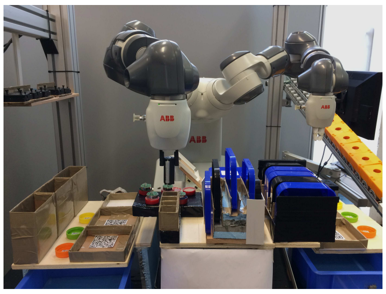
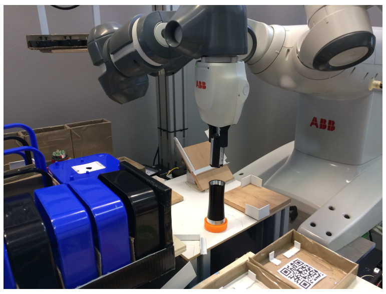

(c)

**Figure 6.20:** Action L1


(a)


(b)


(c)

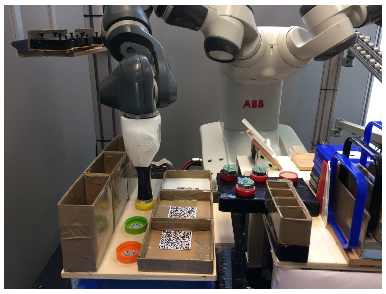
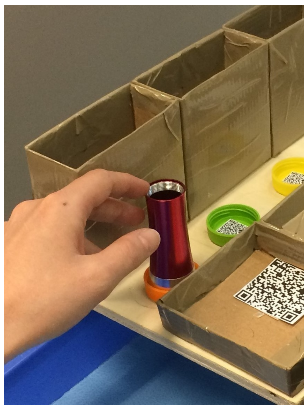
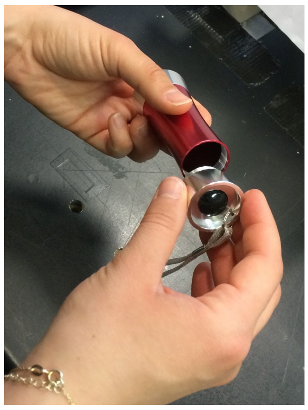

(d)

**Figure 6.21:** Action R2


(a)


(b)

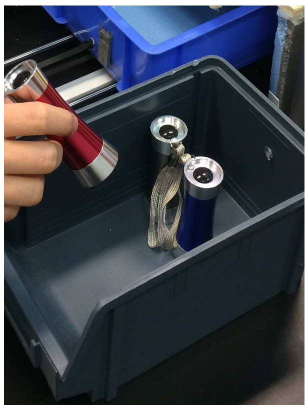
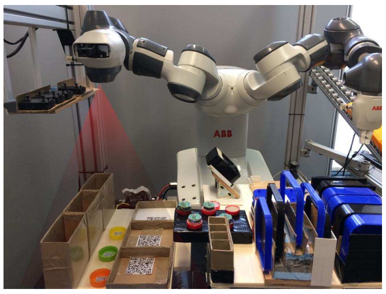
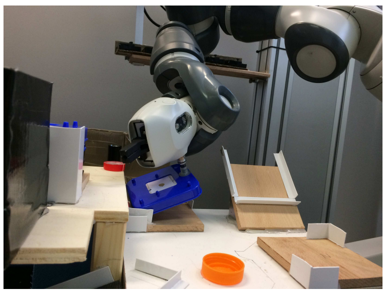

(c)

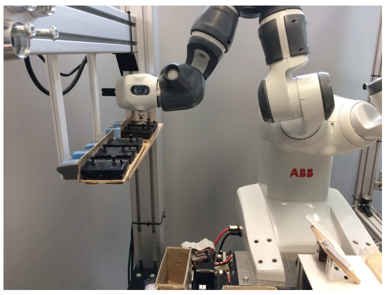
**Figure 6.22:** Action H3

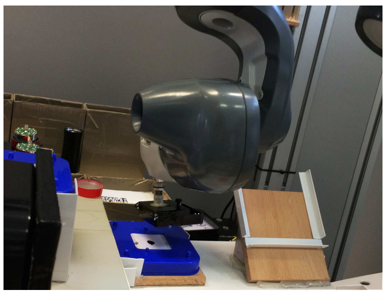

**(a)**

**(b)**

**(c)**


**(d)**


**(e)**


**(f)**

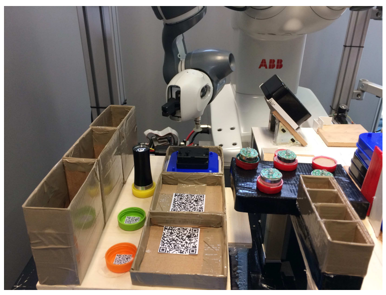
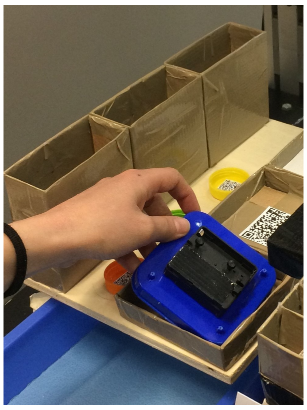
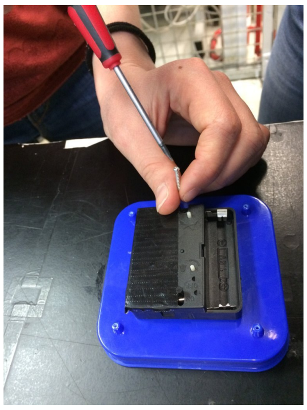

**(g)**

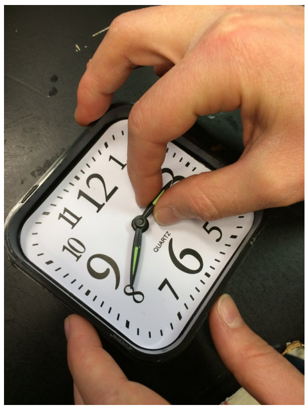
**Figure 6.23:** Action R3



**(a)**



**(b)**

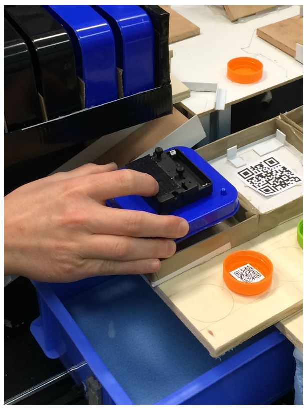


**(c)**

**Figure 6.24:** Action H4 ( part 1 )

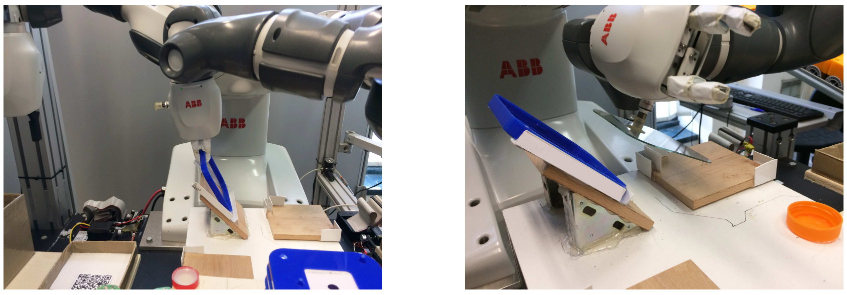
*Figure 6.24: continued on next page.*

*Figure 6.24: continued from previous page.*



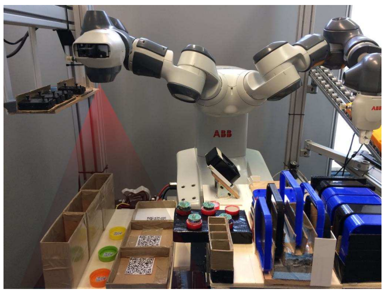
**(d)**

**Figure 6.24:** Action H4 ( part 2 )

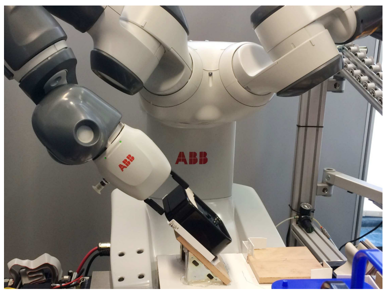


**(a)** **(b)**

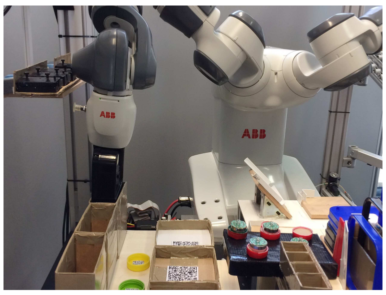
**Figure 6.25:** Action L2



(a)

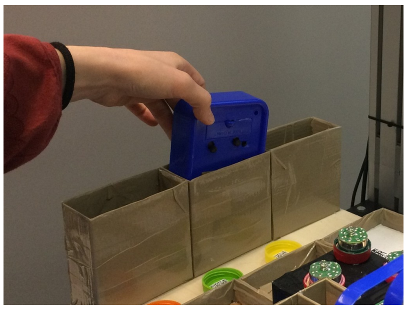


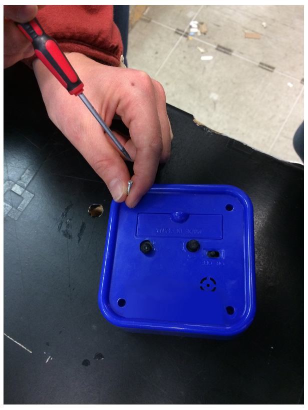
(b)



(c)

**Figure 6.26:** Action R4


(a)


(b)

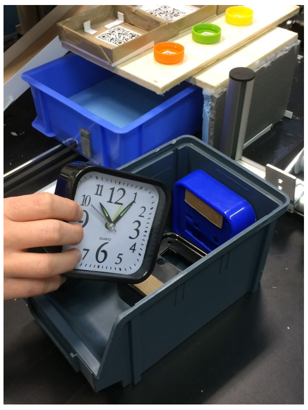

(c)

**Figure 6.27:** Action H5

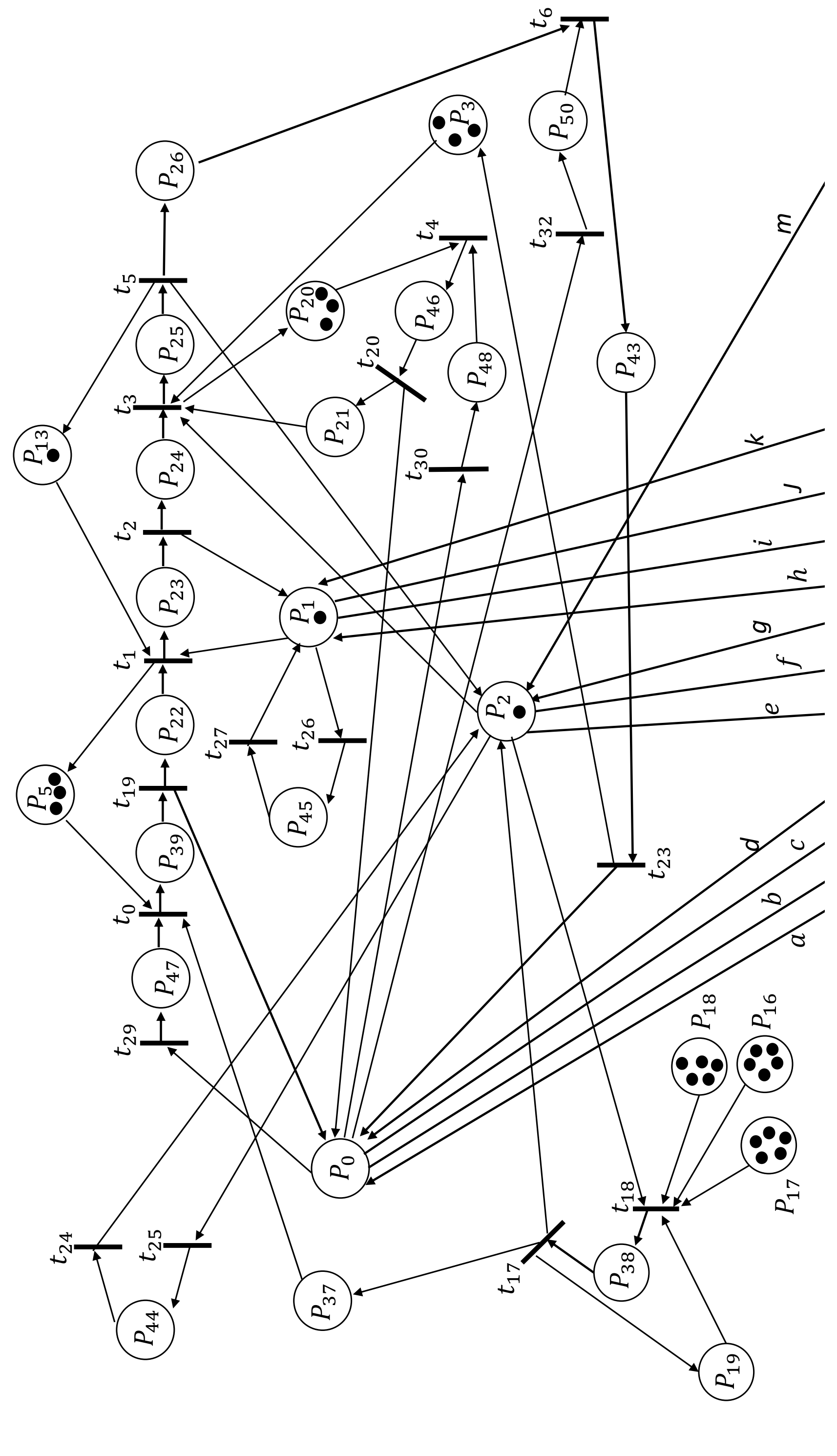


**Figure 6.28:** Fuzzy Time Petri Net part 1 (continued on Figure 6.29)

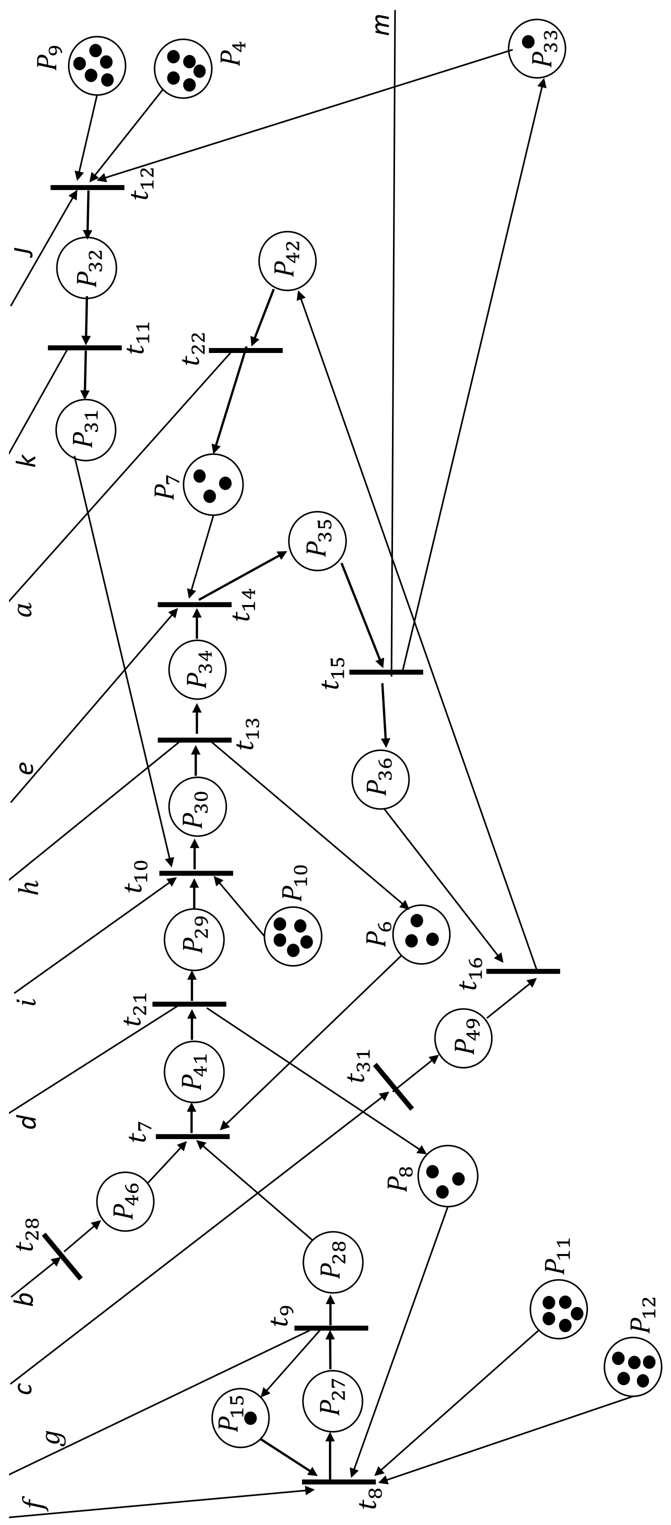


**Figure 6.29:** Fuzzy Time Petri Net part 2 (continued from Figure 6.28)

# Chapter 7

# Conclusions

The aim of this thesis was to develop a scheduling algorithm able to efficiently control a robot involved in a collaboration with a human operator, in an assembly process. In particular, the robot assists the human by providing him or her assembled pieces needed to accomplish his/her tasks.

In this work it has been assumed that the knowledge about the duration of agents' actions was very poor, therefore an approach based on Fuzzy Theory has been adopted to properly model such uncertainty. The system has been represented by a FTPN, in which the robot and the available resources are modeled. In addition, the human collaborator and his/her operations have been included in the FTPN, allowing to obtain a more realistic representation of the overall system. In this way, when generating the reachability tree corresponding to the net, it is possible to represent not only the evolution of the system interpreted as sequence of robot actions, but also to evaluate the incidence of human intentions on such evolution.

To have a robot that promptly satisfies the human's requests, a constantly updated information about the human is necessary: a predictive algorithm that forecasts human's activities is used. Combining the knowledge provided by an MPC approach, the scheduler is able to evaluate all the possible future evolutions of the system, to then choose the most convenient one.
The scheduler acquires information about how the human collaborator behaves, so it learns when he/she is likely to start performing a task. The robotic actions are scheduled accordingly, so that the robot can provide the human with the assembled pieces that he/she will need to continue the process.

Having tested the algorithm on a real use case, where a human and a robot cooperate in an assembly line, the results obtained prove that the scheduler is efficient in terms of computational costs. Moreover, comparing the implemented algorithm with one that does not accurately account for uncertainty, a decrease in the waiting time of agents is obtained, together with a growth in the production rate.

In conclusion, considering all the results, the scheduling algorithm that has been developed in this thesis can be considered a relevant contribution to assembly processes in HRC context.

A possible hint for future studies is the use of different representations of durations of tasks: the approach used in this theory adopts triangular distributions; it is possible to extend it to more complex shapes, such as pentagonal fuzzy numbers, to check if it can further improve the accuracy in describing the knowledge about the agent's tasks.
Moreover, in the scheduling algorithm developed in this thesis, some parameters were set in such a way to be tunable by the user. It would be of interest to introduce more parameters, to be tuned by the human collaborator himself/herself, to obtain a customizable algorithm that better adapts to the specific human operator.
The work developed in this thesis has been applied to a collaboration involving a human and a robot: further studies could deal with a much wider number of agents, i.e. letting more robots and humans cooperate and synchronize with each other, to check how the complexity scales as the dimension of the controlled system increases.

# Appendix

In Section 4.20, the problem of propagating uncertainty was treated and an overview on probabilistic and possibilistic approaches was given. In Chapter 5, the possibilistic approach has been shown in detail, highlighting how possibility distributions propagate throughout a reachability tree and, in particular, Section 5.2.3 treated the method to handle “conflicts”.

In the following, a numerical example to show the propagation of uncertainty with probabilistic approach: it will be seen that a probabilistic conflict involves much more complex computation, supporting the reasons that led to the choice of using possibilistic approach, as stated in Chapter 4.

Consider three Gaussian distributions having probability density functions (*pdf*) as shown on the top of Figure 7.1, having respectively:

$$\begin{array}{cc} \mu_A = 5 & \sigma_A = 0.5 \\ \mu_B = 11 & \sigma_B = 1 \\ \mu_C = 6 & \sigma_C = 0.3 \end{array} \tag{7.1}$$

where $\mu_A$, $\mu_B$, $\mu_C$ are the mean values and $\sigma_A$, $\sigma_B$, $\sigma_C$ are the standard deviations of the related distributions.

Assuming that they are involved in a conflict, the aim is to compute the probability that each of them “wins” against the others.
For instance, the probability that distribution $B$ wins is given by the probability that $B$ takes a specific value $\bar{b}$ that is, at the same time, smaller than $A$ and $C$.

Hence, the probability that $B$ wins is the intersection of three events:

1. $B = \bar{b}$
2. $\bar{b} < A$
3. $\bar{b} < C$

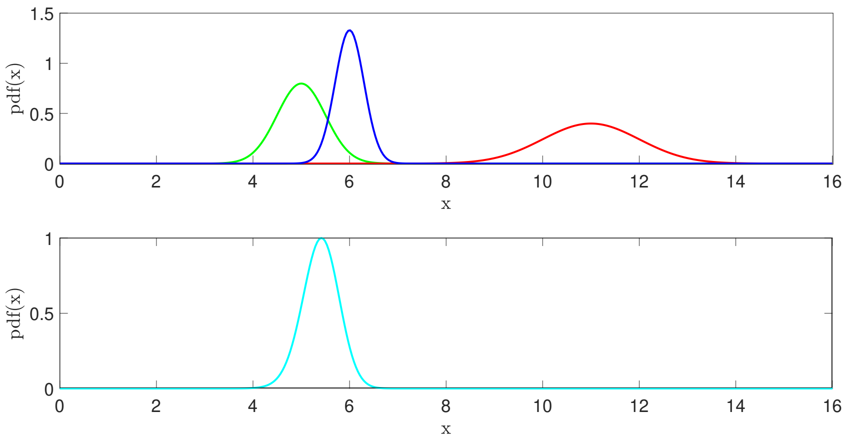


**Figure 7.1:** Top: initial Gaussian distributions; bottom: modified Gaussian distribution, after *conflict*

$\forall \bar{b} \in B$.
Thus, the probability that B is equal to a specific value and the others distributions are greater than such value, is a conditional probability:

$$Prob\_cond\,(\,B\,) = \frac{Prob\,(\,(\,B = \bar{b}\,) \cap (\,\bar{b} < A\,) \cap (\,\bar{b} < C\,)\,)}{\max\,(\,Prob(\,B = \bar{b}\,), Prob(\,\bar{b} < A\,), Prob\,(\,\bar{b} < C\,)\,)}$$

where:

$$Prob\,(\,(\,B = \bar{b}\,) \cap (\,\bar{b} < A\,) \cap (\,\bar{b} < C\,)\,) =$$

$$= Prob\,(\,B = \bar{b}\,) \cdot Prob\,(\,\bar{b} < A\,) \cdot Prob\,(\,\bar{b} < C\,) =$$

$$= Prob\,(\,B = \bar{b}\,) \cdot (\,1 - \int_{-\infty}^{\bar{b}} f_A(x)\,dx\,) \cdot (\,1 - \int_{-\infty}^{\bar{b}} f_C(x)\,dx\,)$$

where $f_A(x)$ and $f_C(x)$ are the cumulative functions (*cdf*) of distributions $A$ and $C$ respectively.
After assuming that B wins the conflict with respect to a single value $\bar{b}$ and after the due computations, the shape of B is conditioned by the other distributions involved in the conflict. As it can be noticed in the bottom of Figure 7.1, the distribution of B is no longer the same the distribution of B in the top of Figure 7.1.

The above computation gave information on the probability that the distributions involved in the conflict are greater than the value tha is

assumed by B.
To compute the overall probability that B wins the conflict, the same procedure should be repeated $\forall \bar{b} \in B$; being the support of B infinite, such computation might not always be solvable in a closed-form.
If this holds, alternative methods and approximations would be needed to solve the integrals, requiring a considerable computational effort (see Section 4.6).

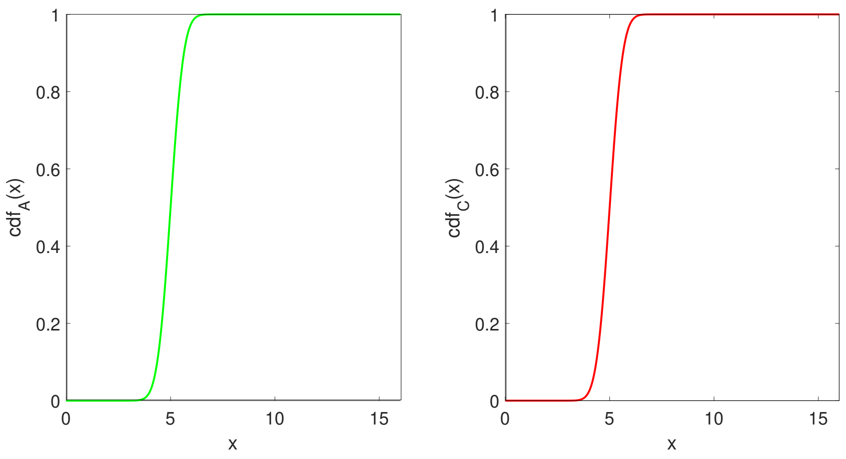


**Figure 7.2:** Left: cumulative function of distribution A; right: cumulative function of distribution C

# Bibliography

[1] Kwon, Woo Young, and Il Hong Suh. *"Planning of proactive behaviors for human–robot cooperative tasks under uncertainty"*. Knowledge-Based Systems 72 (2014): 81-95.

[2] Neumann, Klaus. *"Recent developments in stochastic activity networks"*. INFOR: Information Systems and Operational Research 22.3 (1984): 219-248.

[3] Özdamar, Linet, and Gündüz Ulusoy. *"A survey on the resource-constrained project scheduling problem"*. IIE transactions 27.5 (1995): 574-586.

[4] Herroelen, Willy, Bert De Reyck, and Erik Demeulemeester. *"Resource-constrained project scheduling: a survey of recent developments"*. Computers & Operations Research 25.4 (1998): 279-302.

[5] Herroelen, Willy, Erik Demeulemeester, and Bert De Reyck. *"A classification scheme for project scheduling"*. Project Scheduling. Springer, Boston, MA, 1999. 1-26.

[6] Wang, Juite. *"A fuzzy robust scheduling approach for product development projects"*. European Journal of Operational Research 152.1 (2004): 180-194.

[7] Janak, Stacy L., Xiaoxia Lin, and Christodoulos A. Floudas. *"A new robust optimization approach for scheduling under uncertainty: II. Uncertainty with known probability distribution"*. Computers & chemical engineering 31.3 (2007): 171-195.

[8] Lin, Xiaoxia, Stacy L. Janak, and Christodoulos A. Floudas. *"A new robust optimization approach for scheduling under uncertainty:: I. Bounded uncertainty"*. Computers & chemical engineering 28.6-7 (2004): 1069-1085.

[9] Verderame, Peter M., et al. *“Planning and scheduling under uncertainty: a review across multiple sectors ”*. Industrial & engineering chemistry research 49.9 (2010): 3993-4017.

[10] Elmaghraby, Salah Eldin. *“Activity networks: Project planning and control by network models ”*. John Wiley & Sons, 1977.

[11] Foulds, L., and K. Neumann. *“ Temporal analysis, cost minimization, and the scheduling of projects with stochastic evolution structure ”*. ASIA-PACIFIC J. OPER. RES. 6.2 (1989): 167-191.

[12] Kim, Kee-Eung, and Thomas Dean. *“Solving factored MDPs using non-homogeneous partitionse ”*. Artificial Intelligence 147.1-2 (2003): 225-251.

[13] Guestrin, Carlos, et al. *“Efficient solution algorithms for factored MDPs ”*. Journal of Artificial Intelligence Research 19 (2003): 399-468.

[14] Sutton, Richard S., and Andrew G. Barto. *“Introduction to reinforcement learning ”*. Vol. 135. Cambridge: MIT press, 1998.

[15] Bertsekas, Dimitri P., et al. *“Dynamic programming and optimal control ”*. Vol. 1. No. 2. Belmont, MA: Athena scientific, 1995.

[16] Pineau, Joelle, Geoff Gordon, and Sebastian Thrun. *“Point-based value iteration: An anytime algorithm for POMDPs ”*. IJCAI. Vol. 3. 2003.

[17] Pineau, Joelle, et al. *“ Towards robotic assistants in nursing homes: Challenges and results ”*. Robotics and autonomous systems 42.3-4 (2003): 271-281.

[18] Spaan, Matthijs TJ, and Nikos Vlassis. *“Perseus: Randomized point-based value iteration for POMDPs ”*. Journal of artificial intelligence research 24 (2005): 195-220.

[19] López, María Elena, et al. *“A navigation system for assistant robots using visually augmented POMDPs ”*. Autonomous Robots 19.1 (2005): 67-87.

[20] Sardağ, Alp, and H. Levent Akın. *“ARKAQ-learning: Autonomous state space segmentation and policy generation ”*. International Symposium on Computer and Information Sciences. Springer, Berlin, Heidelberg, 2005.

[21] Porta, Josep M., et al. *"Point-based value iteration for continuous POMDPs"*. Journal of Machine Learning Research 7.Nov (2006): 2329-2367.

[22] Puterman, Martin L. *"Markov decision processes: discrete stochastic dynamic programming"*. John Wiley & Sons, 2014.

[23] Natarajan, Pradeep, and Ramakant Nevatia. *"Coupled hidden semi markov models for activity recognition"*. 2007 IEEE Workshop on Motion and Video Computing (WMVC'07). IEEE, 2007.

[24] Yu, Shun-Zheng. *"Hidden semi-Markov models"*. Artificial intelligence 174.2 (2010): 215-243.

[25] Balasubramanian, Jayanth, and Ignacio E. Grossmann. *"Scheduling to minimize expected completion time in flowshop plants with uncertain processing times"*. Computer Aided Chemical Engineering. Vol. 8. Elsevier, 2000. 79-84.

[26] Balasubramanian, Jayanth, and Ignacio E. Grossmann. *"Scheduling optimization under uncertainty—an alternative approach"*. Computers & Chemical Engineering 27.4 (2003): 469-490.

[27] Petkov, Spas B., and Costas D. Maranas. *"Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty"*. Industrial & engineering chemistry research 36.11 (1997): 4864-4881.

[28] Thompson, Gerald Luther, and Daniel J. Zawack. *"A problem expanding parametric programming method for solving the job shop scheduling problem"*. Annals of Operations Research 4.1 (1985): 327-342.

[29] Li, Zukui, and Marianthi Ierapetritou. *"Process scheduling under uncertainty: Review and challenges"*. Computers & Chemical Engineering 32.4-5 (2008): 715-727.

[30] Li, Zukui, and Marianthi G. Ierapetritou. *"Process scheduling under uncertainty using multiparametric programming"*. AIChE journal 53.12 (2007): 3183-3203.

[31] Li, Zukui, and Marianthi G. Ierapetritou. *"Reactive scheduling using parametric programming"*. AIChE journal 54.10 (2008): 2610-2623.

[32] Ryu, Jun-hyung, and Efstratios N. Pistikopoulos. *"Solving scheduling problems under uncertainty using parametric programming"*. IFAC Proceedings Volumes 34.25 (2001): 203-208.

[33] Ryu, Jun-hyung, Vivek Dua, and Efstratios N. Pistikopoulos. *"Proactive scheduling under uncertainty: A parametric optimization approach"*. Industrial & Engineering Chemistry Research 46.24 (2007): 8044-8049.

[34] Charnes, Abraham, and William W. Cooper. *"Chance-constrained programming"*. Management science 6.1 (1959): 73-79.

[35] Charnes, Abraham, and W. W. Cooper. *"Chance constraints and normal deviates"*. Journal of the American Statistical Association 57.297 (1962): 134-148.

[36] Charnes, Abraham, and William W. Cooper. *"Deterministic equivalents for optimizing and satisficing under chance constraints"*. Operations research 11.1 (1963): 18-39.

[37] Orçun, Seza, I. Kuban Altinel, and Öner Hortaçsu. *"Scheduling of batch processes with operational uncertainties"*. Computers & chemical engineering 20 (1996): S1191-S1196.

[38] Ben-Tal, Aharon, and Arkadi Nemirovski. *"Robust convex optimization"*. Mathematics of operations research 23.4 (1998): 769-805.

[39] Ben-Tal, Aharon, and Arkadi Nemirovski. *"Robust solutions of uncertain linear programs"*. Operations research letters 25.1 (1999): 1-13.

[40] Ben-Tal, Aharon, and Arkadi Nemirovski. *"Robust solutions of linear programming problems contaminated with uncertain data"*. Mathematical programming 88.3 (2000): 411-424.

[41] Ben-Tal, Aharon, et al. *"Adjustable robust solutions of uncertain linear programs"*. Mathematical Programming 99.2 (2004): 351-376.

[42] El Ghaoui, Laurent, and Hervé Lebret. *"Robust solutions to least-squares problems with uncertain data"*. SIAM Journal on matrix analysis and applications 18.4 (1997): 1035-1064.

[43] El Ghaoui, Laurent, Francois Oustry, and Hervé Lebret. *"Robust solutions to uncertain semidefinite programs"*. SIAM Journal on Optimization 9.1 (1998): 33-52.

[44] Bertsimas, Dimitris, and Melvyn Sim. *"Robust discrete optimization and network flows"*. Mathematical programming 98.1-3 (2003): 49-71.

[45] Bertsimas, Dimitris, and Melvyn Sim. *“ The price of robustness ”*. Operations research 52.1 (2004): 35-53.

[46] Bertsimas, Dimitris, Dessislava Pachamanova, and Melvyn Sim. *“ Robust linear optimization under general norms ”*. Operations Research Letters 32.6 (2004): 510-516.

[47] Verderame, Peter M., and Christodoulos A. Floudas. *“ Operational planning of large-scale industrial batch plants under demand due date and amount uncertainty. I. Robust optimization framework ”*. Industrial & engineering chemistry research 48.15 (2009): 7214-7231.

[48] Chanas, Stefan, and Jerzy Kamburowski. *“ The use of fuzzy variables in PERT ”*. Fuzzy sets and systems 5.1 (1981): 11-19.

[49] Hapke, Maciej, and Roman Slowinski. *“ Fuzzy priority heuristics for project scheduling ”*. Fuzzy sets and systems 83.3 (1996): 291-299.

[50] Wang, Li-Xin. *“ Analysis and design of hierarchical fuzzy systems ”*. IEEE Transactions on Fuzzy systems 7.5 (1999): 617-624.

[51] Zanchettin, A., Casalino, A., Piroddi, L., & Rocco, P. (2018). *“ Prediction of human activity patterns for human-robot collaborative assembly tasks ”*. IEEE Transactions on Industrial Informatics.

[52] Petri, Carl Adam. *“ Introduction to general net theory ”*. Net theory and applications. Springer, Berlin, Heidelberg, 1980. 1-19.

[53] Liverani, Marco. Appunti sulle Reti di Petri. Maggio 2004.

[54] Silva, José Reinaldo, and Pedro MG del Foyo. *“ Timed petri nets ”*. Petri Nets-Manufacturing and Computer Science. IntechOpen, 2012.

[55] Mura, Ivan, and Andrea Bondavalli. "Markov regenerative stochastic Petri nets to model and evaluate phased mission systems dependability." IEEE Transactions on Computers 12 (2001): 1337-1351.

[56] Kurkovsky, Stanislav, and Rasiah Loganantharaj. *“ Extension of Petri nets for representing and reasoning with tasks with imprecise durations ”*. Applied Intelligence 23.2 (2005): 97-108.

[57] Cividini, Filippo. *“ A scheduling algorithm for human-robot collaborative assembly tasks ”*. (2017).

[58] Shannon, Claude Elwood. *“A mathematical theory of communication ”.* Bell system technical journal 27.3 (1948): 379-423.

[59] Shannon, Claude E., Warren Weaver, and Arthur W. Burks. *“The mathematical theory of communication ”.* (1951).

[60] Weaver, Warren. *“The mathematics of communication ”.* Scientific American 181.1 (1949): 11-15.

[61] Seising, Rudolf. *“60 years, A Mathematical Theory of Communication - Towards a Fuzzy Information Theory ”.* IFSA/EUSFLAT Conf. 2009.

[62] Li, Yiping, Jianwen Chen, and Ling Feng. *“Dealing with uncertainty: A survey of theories and practices ”.* IEEE Transactions on Knowledge and Data Engineering 25.11 (2013): 2463-2482.

[63] Zadeh, Lotfi Asker. *“Fuzzy sets as a basis for a theory of possibility ”.* Fuzzy sets and systems 1.1 (1978): 3-28.

[64] Baraldi, Piero, and Enrico Zio. *“A combined Monte Carlo and possibilistic approach to uncertainty propagation in event tree analysis ”.* Risk Analysis: An International Journal 28.5 (2008): 1309-1326.

[65] D. Dubois, H. Prade, *“Possibility Theory: An Approach to Computerized Processing of Uncertainty ”*, Plenum Press, NewYork, 1988.

[66] Cunha, Americo. *“ “Modeling and quantification of physical systems uncertainties in a probabilistic framework ”.* Probabilistic Prognostics and Health Management of Energy Systems. Springer, Cham, 2017. 127-156.

[67] Nahmias, Steven. *“Fuzzy variables.“ Fuzzy sets and systems ”* 1.2 (1978): 97-110.

[68] Jager, Rene. *“Fuzzy logic in control ”.* Rene Jager, 1995.

[69] Iqbal, Muhammad Sarfraz, and Ullrika Sahlin. *“Treatment of Epistemic Uncertainty in Environmental Fate Models–Consequences on Chemical Safety Regulatory Strategies ”.* PSAM11 & ESREL 2012, Helsinki, Finland, 25-29 June 2012. 2012.

[70] Yaghobi, M., M. Rabbani, and M. Adabitabar Firozja3 J. Vahidi. *“Comparison of Fuzzy Numbers with Ranking Fuzzy and Real Number ”.*

[71] Pirzada, U. M., and D. C. Vakaskar. *"Existence of Hukuhara differentiability of fuzzy-valued functions"*. arXiv preprint arXiv:1609.04748 (2016).

[72] Amini, Amin, and Navid Nikraz. *"A Method for Constructing Non-Isosceles Triangular Fuzzy Numbers using Frequency Histogram and Statistical Parameters"*. Soft Computing in Civil Engineering 1.1 (2017): 65-85.

[73] Cividini, Filippo. *"A scheduling algorithm for human-robot collaborative assembly tasks"*. (2017).

[74] de Barros, Laécio Carvalho, Rodney Carlos Bassanezi, and Weldon Alexander Lodwick. *"The extension principle of zadeh and fuzzy numbers"*. A First Course in Fuzzy Logic, Fuzzy Dynamical Systems, and Biomathematics. Springer, Berlin, Heidelberg, 2017. 23-41.

[75] Lazzerini, Beatrice. *"Introduzione agli insiemi fuzzy e alla logica fuzzy"*. dipartimento di ingegneria dell'informazione, università di Pisa (2000).

[76] https://ekoharsono.files.wordpress.com/2013/01/5-fuzzy-numbers-arithmetic-and-the-extension-principle.pdf