On Planning in Human-Robot Collaboration

Guilherme de Mascarenhas Belo Antunes
guilherme.mascarenhas@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

***Abstract*—As robots leave highly structured factory environments and enter human-populated areas, autonomous systems should learn how to adapt and cooperate with humans. The key to designing such systems is accurate action prediction. Previous human action prediction research focused predominantly on two approaches: 1) collecting data about a given problem and making the algorithm "learn" its inherent patterns (neural networks); 2) describing the problem's rules and constraints and letting the algorithm find a sequence of actions that takes us from a given initial state to a goal state (planning). The first approach completely ignores the structure of the problem. Additionally, its heavy data dependency makes this strategy hard to employ when little to no information is available. On the other hand, the second approach struggles to find an action sequence when the problem is too complex. This work proposes a model-based reinforcement learning strategy, combining the previously mentioned approaches. By describing the rules of the environment to the algorithm, it learns what actions the robot agent and the human agent can carry out. It then estimates what option brings the most joint reward to both entities with the help of a policy network. We implemented multiple models to study how knowledge about future possibilities can increase the cooperative behavior of the robot agent. Our work shows that focusing on an agent's objective may lead to better human-robot collaboration than targeting the human's next move.**

***Index Terms*—Planning, Joint action, Prediction, Planning heuristics**

I. Introduction

With the growing number of autonomous systems leaving structured environments, they must be able to adapt and collaborate with humans. Collaborative robots (or cobots) are machines specifically designed for close human-robot interaction within a shared physical space. Human-robot collaboration's applications extend throughout multiple areas such as autonomous driving [1], healthcare [2], risk assessment [3], [4], among others.

To achieve collaboration, robots must learn how to act like humans, which in this context, means predicting and adapting to others' behavior.

As people, we rely on verbal communication and, more importantly, non-verbal cues to understand each other's goals and coordinate [5]. If we were to give this "social intuition" to autonomous systems, combined with their precision, they would be able to perform human tasks better than humans.

In this thesis, we focus on the importance of action prediction as a tool to achieve collaboration. Human behavior is not always predictable, as humans often act according to their individual beliefs and experience. This fact makes it harder for machines to anticipate our actions accurately.

The objective here is to create an artificial agent that can collaborate with humans by understanding human intentions and predicting their future actions.

II. Collaboration

To understand how we can program robots to collaborate with humans, we must first understand how humans can collaborate among themselves.

Research shows that signs of the ability to coordinate with others are present in humans from an early age. Through interaction with other people, children become capable of engaging in more complex joint actions [6].

Sebanz, Natalie, et al. define cooperation as the interaction of three components: shared representation, action prediction, and anticipative action [7]. Shared representation relates to establishing a common goal and structure for the joint action. Action prediction is related to predicting the co-actors' next move. Anticipative action, in turn, involves acting according to the goal and our predictions of the other agents' behaviors.

In order to predict an agent's action we need to comprehen what is guiding it. The difficulty here is understanding how automated machines can "assign" a goal to an agent.

The philosopher Daniel Dennet introduced the concept of intentionality, a core component of the cognitive science field. Dennet states that actions are goal-directed and derive from unique beliefs and desires [8]. This collection of psychological features is what we call a mental state.

A. Theory of Mind

The Theory of Mind mechanism (ToMM) is an essential field of research in understanding and reasoning about goal-directed entities. This mechanism refers to the ability underlying the capacity to reason about one's own and others' mental states. Possessing a functional theory of mind is crucial for successful everyday social interactions, as it allows us to make sense of the actions of others and communicate efficiently.

If we were able to design systems that are capable of understanding the intents of others and attributing mental states to them, i.e., desires, thoughts, or intentions, we would achieve better human-robot cooperation.

In order to understand how humans come to develop a theory of mind, authors like Simon Baron-Cohen and Alan M. Leslie have focused on the study of children on the autism spectrum. [9], [10]

Previous studies had suggested that these children were unable to comprehend other people's mental state. The false

belief test clearly illustrates this idea. In this test, a child observes the scenario presented in figure 1.

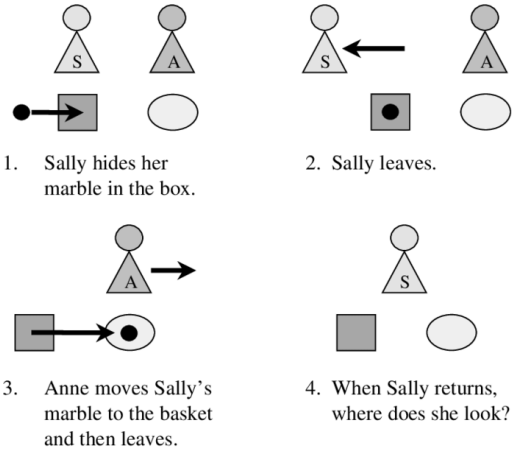

Fig. 1. False Belief test from Brooks, Rodney et al. [11]

The experiment demonstrated that children around a certain age were able to understand that when Sally returned she would look for the marble in the box, and therefore hold a false belief. However, children on the autism spectrum around the same age thought Sally would look for the marble in the basket, since they could not comprehend that other people have their own beliefs (mental states).

This experience raises the question: How limited is the perception that children on the autism spectrum have? By answering this question, we hope to explain theory of mind a bit further.

Alan M. Leslie proposes a distinction in the development of the normal functioning cognitive system, between primary representation and metarepresentation [12]. Primary representations allow humans to understand and store literal information about the world. They refer to simple predicates like "There is a snake in the bush". Metarepresentations, in contrast, allow our cognitive system to create descriptions of other representations, hence their name, e.g.,"Bob thinks there is a snake in the bush". According to Leslie, metarepresentation is the mechanism that enables humans to represent mental states.

When it comes to children on the autism spectrum, research has shown that their primary representation mechanism is intact, but the metarepresentation one is not. This conclusion comes from observations where joint attention activities (e.g., pointing at an object to get someone to hand it to you) are considerably less common in children on the autism spectrum than in control groups [13].

Does this mean that joint attention behaviors require metarepresentation? Some authors seem to agree that the answer is true [9], [14]. The consensus is that collaboration requires communication, whether it is verbal or non-verbal (e.g., gaze or pointing). Communication (i.e., each person understanding the intents of the other) requires each individual to have the theory of mind mechanism. Theory of mind needs metarepresentation. Therefore, we can conclude that joint attention requires metarepresentation.

B. Non-verbal communication

Thus far, we have omitted the role of communication in joint actions. Humans may use verbal conversation to agree on the objective and strategy they will employ. However, the majority of communication will be at a non-verbal level. Giving small signals to our co-actor while performing a joint action is part of human nature.

Non-verbal cues can take many forms, e.g., facial expressions, gestures, paralinguistics, and more.

The eye gaze, in particular, plays a central role. Humans use actions like staring and blinking to transmit a ton of information.

Eyes are a unique tool. While ears allow us to hear and lips enable us to speak, eyes can both signal and perceive [15], [16].

Signaling and perceiving are always hand in hand. If we imagine a social context where one person signals something, others perceive it (figure 2).

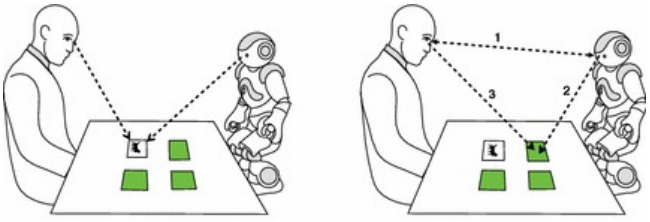

Fig. 2. Directing attention through gaze hints

Huang, Chien-Ming, et al. directed their efforts to examine gaze patterns to predict intentions [17]. Their exercise considers a sandwich-making scenario where a customer requests ingredients, and the worker adds them to the sandwich. They verified that, generally, the customer would look at an item before asking for it. The results lead us to the conclusion that eye gaze manifests an intention. This idea extends to other forms of non-verbal cues.

Here, we focus on information like distance to objects or direction of movement, which are more suitable for the spatial nature of the problems we intend to solve.

C. Theory of mind for Robots

So far, we have looked at how the theory of mind mechanism helps people assign mental states to each other.

If we were building machines that interact naturally with people, they must interpret both the environment and the animate agents. They must also display social cues so that co-actors can predict their moves.

To implement the theory of mind mechanism in robots, we consider Baron-Cohen's model and its implications (fig 3).

This model considers two forms of input. Firstly, we have the raw perceptions that come from our senses (e.g., vision, hearing, touch). The intentionality detector (ID) filters this data and leaves only self-launched actions, i.e., actions that have an

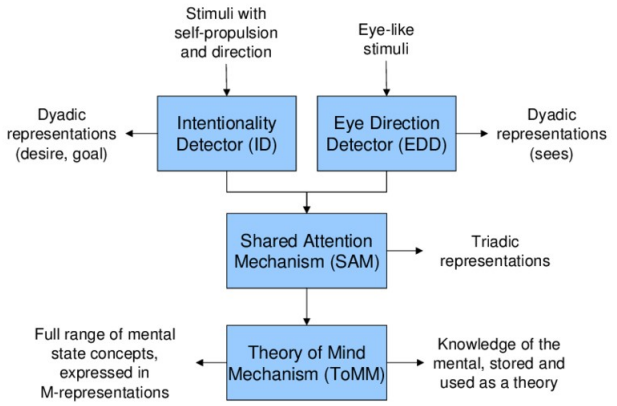

Fig. 3. Block diagram of Baron-Cohen's model of the development of theory of mind [18].

intention behind them. These motions allow it to distinguish between animate (agents) and inanimate (objects) entities [19]. The ID module then generates representations of the partner's intentions, e.g., while looking at someone lifting a sofa, "He is doing it alone" or "He is waiting for something".

The second set of inputs are visual stimuli. The eye direction detector (EDD) builds representations specifying if the gaze of another agent is concentrating on you. This module then identifies if you are the target of that agent's attention [20].

The third module, the shared attention mechanism (SAM), combines the representations of ID and EDD to create more complex portrayals. Imagine John is looking at us while he tries to lift the sofa. We build descriptions such as "John knows that I see him looking at me" or something along those lines.

The fourth and last module, the theory of mind mechanism (ToMM), is what transforms SAM representations into mental states [21]. For example, by combining "John is trying to lift the couch", "John is looking at me", and "John knows that I see him looking at me", we can get "Jonh wants my help lifting the couch".

The difficulty in implementing this model is that every module requires analyzing several complex processes.

There are numerous works in mental state attribution and gaze following that provide frameworks for the modules presented above. However, we won't look at them in depth as they are outside the scope of this work.

Nonetheless, the model described in this section highlights some valuable ideas for modulating human behavior. We utilize them in an algorithm to predict actions.

In the following sections, we describe techniques that achieve collaboration. One group of methods solely focuses on predicting the next move, while the other tries to understand the steps that compose each action.

III. Prediction

Focusing on prediction, as opposed to focusing on the goal, suggests that the predictive system does not know the global structure of the environment. Instead, the algorithm analyzes samples, i.e., data about previous iterations of the problem, and gives a prediction.

This corresponds to, e.g., choosing the best move in a chess game without knowing the rules but having seen the game multiple times before. The concept of playing chess without knowing the rules previously may seem empirically wrong. However, several authors applied it, achieving convincing results. [22], [23]

These approaches save us the trouble of formulating the environment's rules, which implies we can reuse them in several scenarios with relatively little effort.

Getting predictions by studying data introduces the machine-learning field of artificial neural networks (ANNs or NNs).

A. Artificial Neural Networks

Neural networks (same as artificial neural networks) are algorithms inspired by the neurons in the biological brain. They can recognize patterns and correlations in data and use them to build a classification model.

NNs are composed of several "artificial neurons" with connections between them and divided into three layer types: input layer, hidden layers, and output layer.

Neurons connect to neurons in the adjacent layers through weighted edges. This way, they serve as inputs to neurons in the next layer. The weight represents the connection's strength, i.e., how strongly one neuron affects the other. The neurons' layout is summarized in figure 4.

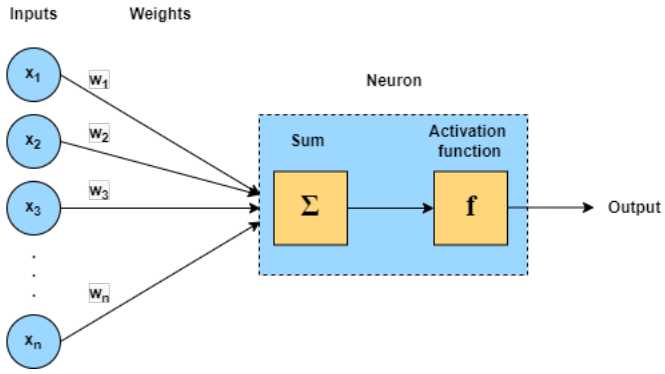

Fig. 4. Artificial neuron's layout

The network learns patterns in a process called training. In this process, we take data samples from our problem (inputs and corresponding outputs) and feed them to the network. The algorithm repetitively evaluates how wrong its predictions are and corrects the edges' weights to make them more accurate.

In the context of action prediction, the commonly used models have slightly different architectures.

B. Convolutional Neural Networks

Convolutional neural networks (CNNs) are the most commonly applied model in image analysis/recognition. Their success comes from the ability to identify patterns (or features) in images.

The first big innovation is the introduction of convolutional layers.These layers are the ones responsible for highlighting features. They get their name from the convolution operation they perform.

The first convolutional layers will capture low-level details like edges and corners. As we progress on the network, we get more high-leveled features, e.g., if the image contains a dog.

The second novelty is the pooling layers. Usually, these layers always follow a convolutional layer. Their purpose is to reduce the size of the convolved feature. This way, they reduce the computational power required for processing the data. Additionally, they also extract dominant features formerly highlighted by the kernels. The most common operation performed by pooling layers is "max pooling". This operation calculates the maximum value in a convolved feature patch.

The computer vision field utilizes CNNs in one of two ways: 1) processing an action within a video and generating a label (action recognition); 2) predicting a human action from a temporally incomplete video (action prediction).

Due to the growing real-world applications, these problems have become more popular [24]. Following, we will describe relevant research in each field.

Security surveillance is a frequent application of action recognition. Areas under surveillance often prohibit some human actions, e.g., jumping over a fence.

Ji et al. developed a 3D CNN model that extracts spatial and temporal features by applying 3D convolutions [25]. This way, the model captures motion information in sequential frames and labels the video.

Duan et al. study the use of "skeleton-based" approaches in action recognition. Their model estimates the pose in a 2D human image. Then, they stack heatmaps of joints or limbs along several frames, creating 3D heatmaps. Finally, a 3D-CNN classifies these 3D heatmaps.

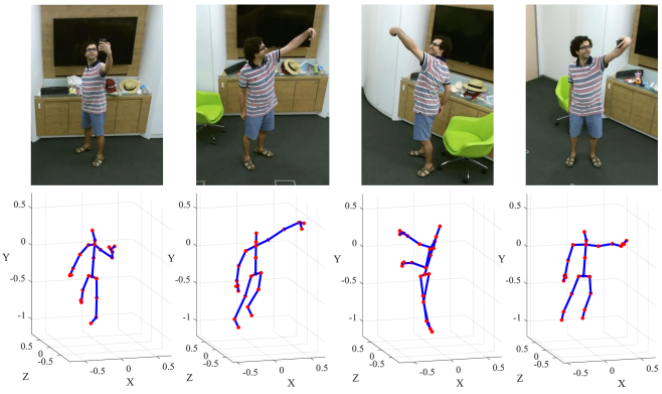

Fig. 5. Skeleton representations in human action frames

Gkioxari et al. introduce the idea of using cues in an image to classify the action [26]. They argue that contextual information may help improve the performance of action recognition models. For example, when observing a jogger, the scenario (e.g., the road or trail) and the presence of other joggers can be an additional source of information. Their model outperforms previous approaches in the field.

Correctly classifying an act, watching only its initial frames, is crucial in applications like autonomous driving. By analyzing the body motion and predicting the future actions of pedestrians, CNNs can prevent collisions.

Kong et al. propose a model for predicting actions in videos containing incomplete acts [27]. Their approach uses sequential context information to complement the extracted features of partial videos. It reconstructs missing info by learning from fully observed videos of the corresponding action.

CNNs are considerably convenient when it comes to raw image classification. However, to identify patterns in sequential data, such as actions, there is another class of neural networks that stands out.

C. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are a class of neural networks that allow previous outputs to be used as inputs. The main difference between a CNN and an RNN is the ability to process temporal information.

This capacity makes them very appropriate in the fields of speech recognition and natural language processing. Data in these fields come in sequences, e.g., a sentence. As described in the previous section, we can represent an action as a collection of sequential frames. In the case of speech recognition, the problem might be predicting the succeeding words in a sentence, whereas in action prediction, we can use RNNs to anticipate the next set of frames.

The ability to recognize patterns in sequential data and predict the most likely output comes from their architecture. Utilizing previous results to elaborate the current prediction makes RNN cells differ from simple neural network cells (figure 6).

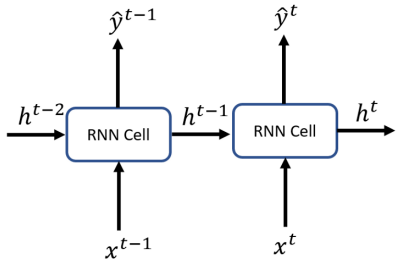

Fig. 6. RNN cell sequence

RNN cells receive the input vector x^t, where t is the time, and projects it to an output vector $\hat{y}^t$. As we can observe in figure 6, to make a prediction, the cells also consider what is called the network hidden state, h_{t-1} (hidden state in the previous timestep), which depends on the previous outputs of the network.

At each timestep, the RNN calculates the output vector and the hidden state as a some function of the input and the

previous hidden state. It then propagates the current hidden state to the next cell.

In short, an RNN cell is defined by an expression for the output at the current timestep, $\hat{y}^t$, and the next internal state, h_t (equation 1). Here W_x, U_x, and b_x are model parameters.

$$\begin{aligned} h_t &= softmax(W_h h_{t-1} + U_h x_t + b_h) \\ \hat{y}_t &= softmax(W_{\hat{y}} h_t + U_{\hat{y}} x_t + b_y^t) \end{aligned} \tag{1}$$

The shortcoming of RNNs is they cannot remember long-term dependencies in data due to the vanishing gradient problem. To solve this, Sepp Hochreiter and Jurgen Schmidhuber invented Long Short-Term Memory (LSTM) [28]. RNNs built with LSTM cells divide data into short-term and long-term categories. RNNs choose whether data is valuable enough to remember and loop back into the network or if it is irrelevant.

LSTM cells consist of three parts, also known as gates: the forget gate, the input gate, and the output gate (figure 7).

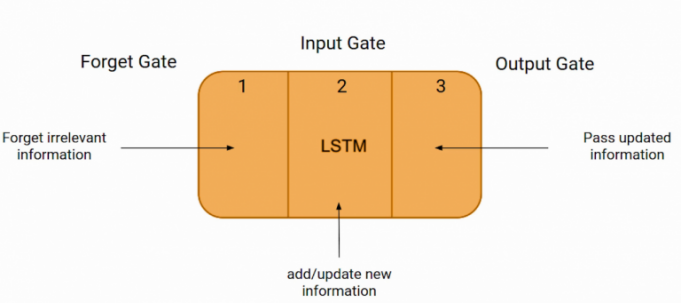

Fig. 7. Long short-term memory cell [29]

By regulating the flow of information into and out of the cell, the three gates allow the RNN to have a short-term memory that can last several time steps, thus the name long short-term memory.

Graves et al. combined a multidemensional LSTM with temporal classification to create a handwriting recogniser [30]. They were able to successfully apply their system to English and Arabic. Additionally, they anticipate that this model can be used for any supervised sequence labelling problem.

In another area, Seker et al. showed that it is possible to use RNNs for condition monitoring and diagnosis in nuclear power plant systems [31]. By comprehending specific patterns in data, the network can detect anomalies. i.e., results that don't correspond to the prediction. The conclusions of this study make us speculate about utilizing similar techniques to detect human errors.

In the context of action prediction, the most well-known example is autonomous driving. Rasoule et al. introduce a model that collects information from various sources to accomplish pedestrian prediction at the point of crossing [32]. At each time step, the model observes the pedestrian and its surroundings, which allows it to perceive its pose and velocity. They prove that the algorithm outperforms previous architectures, which used motion history to predict future pedestrian trajectories.

Accurate driver action prediction can improve driver assistance systems. Early recognition is critical in this scenario, as it can help detect driver mistakes and prevent accidents. Olabiyi et al. present a model that captures information about the driver and the driving environment with the assistance of cameras [33]. They then use RNNs to determine the correlation between the inputs and the driver's behavior. Their algorithm can predict actions like accelerating, braking, and lane changing a considerable time before it happens.

Data-centered approaches have some limitations, the first one being heavy data dependency. NNs use considerable amounts of data in the training process. This principle prevents us from using NNs in environments where little to no information is available.

The second limitation is overfitting. This concept refers to when a statistical model fits exactly the training data. While training NNs, we want the algorithm to learn patterns in this training set and then accurately perform with unseen data. If the model perfectly adapts to samples, it loses its ability to generalize for new inputs.

Following this we introduce a different paradigm, reinforcement learning.

D. Reinforcement Learning

Reinforcement Learning (RL) is an area of machine learning that focuses on choosing the action that maximizes a reward. It differs from regular neural networks as it learns from experience, i.e., trial and error, in an interactive environment.

Reinforcement learning is usually modeled as a Markov Decision Process (MDP), which we will better describe later. In this model, the agents interact with the environment through actions. The environment returns a state and a reward for that corresponding action (8). Randomly selecting moves and observing the environment's response allows the model to discover what actions are better. This process is equivalent to training in neural networks. The purpose is for the agent to learn an optimal policy that maximizes the expected cumulative reward of a sequence of actions.

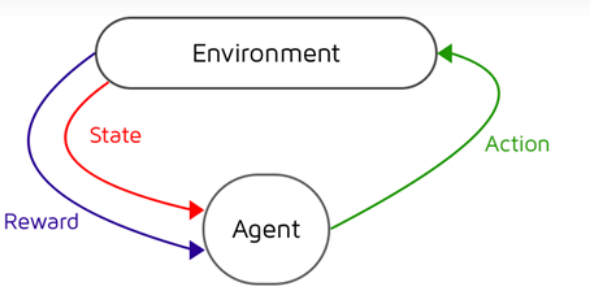

Fig. 8. Reinforcement learning model

IV. Action

The previous section described a set of approaches that use data samples about the problem to learn how to predict future cases. We now focus on a method that utilizes the rules of the environment to predict future states.

A. Planning

AI planning is a field of Artificial Intelligence (AI) that explores the process of using autonomous techniques to solve planning and scheduling problems [34]. A planning problem is one in which we have some initial state, which we want to transform into the desired goal state by applying a set of actions. AI planning performs something known as a state space search. If we consider the initial state as the root, we can successively apply actions to get a tree of possible states. Figure 9 provides a visual example of this search.

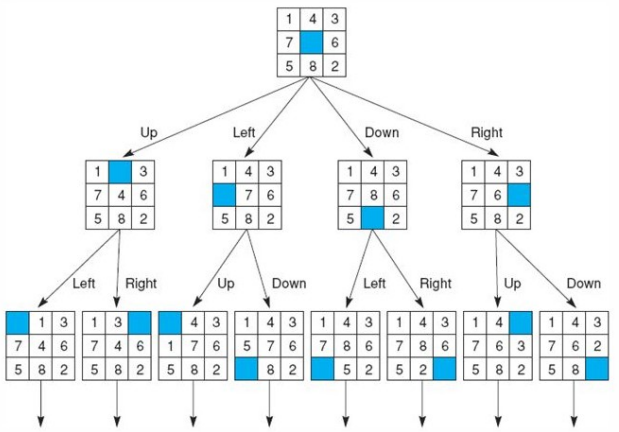

Fig. 9. Example of state space search in 8-puzzle

B. Algorithms

The algorithm we choose influences the search's performance. Classical planning problems use algorithms that fall into two classes: uninformed and informed (or heuristic).

Uninformed search methods (also called blind methods) are an approach to state space searches where the algorithm uses brute force operations to find a solution. In a given state, there are a set of possible actions. The algorithm blindly selects one, regardless of whether it is a good option.

We highlight two uninformed search algorithms: Depth-first search (DFS) and Breadth-first search (BFS).

Uninformed search techniques are appropriate when we have no information on the problem. However, artificial intelligence often requires more refined mechanisms.

Informed search is a class of algorithms that uses problem-specific knowledge about the domain to achieve a more efficient search. These methods guide the search toward more promising paths. We focus on a specific algorithm, the "A-star" search. Unlike other traversal techniques, A* (A-star) uses information on the goal state which helps with planning ahead at each step so a more optimal decision is made.

A* expands paths that are less expensive by using the function:

$$f(n) = g(n) + h(n), \tag{2}$$

where

- $f(n)$ = total estimated cost of path through node n
- $g(n)$ = cost so far to reach node n
- $h(n)$ = estimated cost from node n to goal (heuristic)

C. Combined Approaches

In the context of sequential decision-making, there are two approaches: reinforcement learning and planning. In this section, we focus on a single application of the combination of both strategies. AlphaGo is a computer program that plays the board game Go. It introduces a combination of advanced (heuristic) tree search techniques with deep neural networks. The Monte-Carlo tree search (MCTS) is a heuristic search algorithm that uses Monte-Carlo rollouts (simulations) to estimate the reward of each state. The neural networks take a description of the Go board as an input and process it. One neural network, the "policy network" selects the next move to play. The other neural, the "value network", predicts the winner of the game. These networks are trained, firstly, by data from human expert games, and secondly, by games of self-play (reinforcement learning) [35].

V. Proposed Model

Human-robot collaboration requires the robot to anticipate human action. This action anticipation relates to predicting possible future actions and selecting the most probable ones with the help of environmental cues. We model this problem as a Markov Decision Process (MDP).

A. Problem Statement

The MDP is a model for decision-making scenarios where the output depends on the decision maker's behavior and some random conditions (stochastic model). In our case, the decision-maker (AI agent) tries to select the action that brings the most reward, considering the human's possible actions (random condition).

A MDP is a tuple $< S, A, P_a, R_a >$ with:

- Set of possible environment states (state space), S
- Set of possible agent actions, A
- Set of state transition probabilities, P_a. Here $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$.
- Expected immediate reward function, R_a. The reward after transitioning from state s to state s' by choosing action a is represented by $R_a(s, s')$

To test the implementation of our model in a context of action prediction, we defined a relatively simple domain with the following characteristics:

- **Environment** - 5x5 (2D) grid world.
- **Objective** - Agents need to collect items scattered around the grid and deliver them at a goal location.
- **Number of Agents** - 1 or 2 agents (AI agent + human agent).
- **Goal location** - A set of coordinates, e.g., goal location $= (2, 2)$
- **Items** - Several vegetables scattered around the grid.
- **Possible Actions** - Agents can move up, down, left, or right. If they are at the location of an item, they can pick it up. If they are at the goal location they can put it down. Agents can only hold one item at a time.

B. Model's Approach

Our model's approach reutilizes some "AlphaGo concepts" in the context of human action prediction. We define an initial state and a goal state as classical planning problems do. To ensure the algorithm follows the laws that regulate our domain, we provide some information that allows it to assert possible actions. This way, it doesn't consider impossible moves and gains the ability to plan several timesteps ahead. We can see this description as an attempt to give the algorithm a shared representation.

With this information, the algorithm predicts both the user's possible moves and the AI agent's possible moves.

Then, the action selection process will evaluate what action combination brings the most shared reward to our agents based on a policy. This policy will ideally be the result of training a reinforcement learning model. The algorithm utilizes this advanced tree search method to move an AI agent while the human acts. Figure 10 presents the model's diagram.

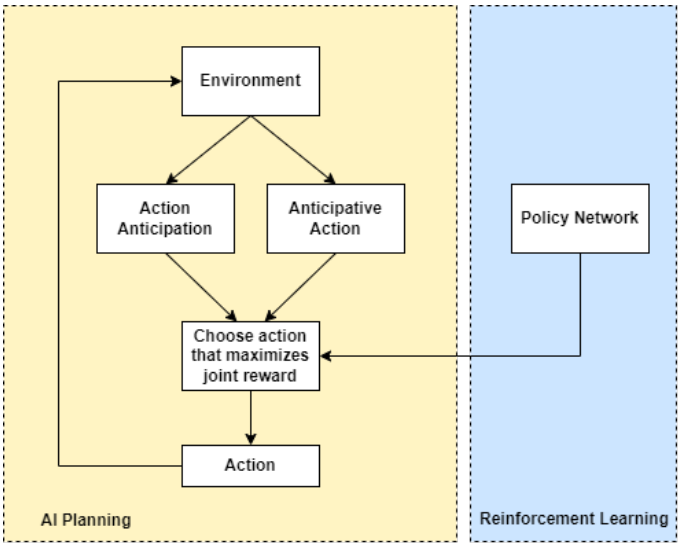

Fig. 10. Proposed model diagram

C. Implementation

The objective of this work was to create an AI agent that can collaborate with a human agent (user). We described that one of the conditions for cooperation is that both agents must know the goal and what they can do to achieve it. This way, our first model is an agent that could complete the task alone.

1) Single Agent model: The single agent model, although relatively much more simple than our initially proposed model, requires us to define most of the logic present in the domain. We start by implementing an agent that acts optimally.

Since we want to find a minimal-cost path, the algorithm consecutively explores positions with the lowest total cost estimate. This way, it provides us a plan (sequence of actions from the initial state to the goal state).

Now that we have an AI Agent capable of solving the problem on its own, we can insert it in a shared physical space.

2) Simple Multi-Agent model: If we implement no modification to the single-agent model algorithm, the agent will be unaware of the human's presence and the impact of human actions in the environment. To achieve cooperation, we need the agent to predict what actions the human will perform and adapt if the prediction is wrong. After all, as Grosz says, "Collaboration is not just sums of individual plans" [36].

3) Replanning Agent model: In this model, the AI understands that the relationship between the environment and each agent is fundamentally the same. In other words, the algorithm considers that the other agent can also change the state of the shared physical space. We can look at this as some form of Theory of Mind. Figure 11 displays this model's simplified architecture.

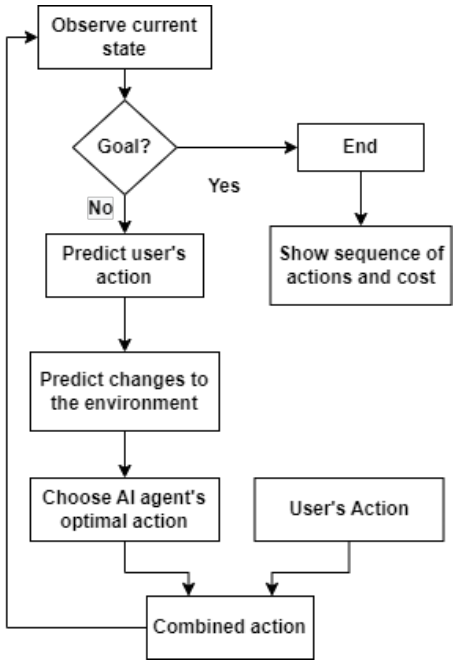

Fig. 11. Replanning Agent algorithm's structure

Replanning allows the agent to adapt to changes in the environment. However, this model still has some imperfections. Firstly, predicting the user's move only relies on the first solution found. The algorithm must consider that there might be more than one optimal solution to become more accurate. Secondly, the AI agent knows that the other agent can interact with the environment but doesn't know it has a mind of its own. The human might be closer to an object than the AI agent, but the AI still tries to move to that location.

4) Goal-Predicting Agent model: This approach gets rid of the previous limitations since it considers that the other agent (human) has a mind of its own, and each move is, therefore, part of a plan to achieve a given objective.

At the beginning of the problem execution, we enumerate the sub-tasks that compose the final goal. In a real-world

application, this would be like having a recipe. Each agent selects a sub-task from the list according to a strategy, e.g., complete tasks that take longer first.

We present the model for the AI agent's behavior in figure 12.

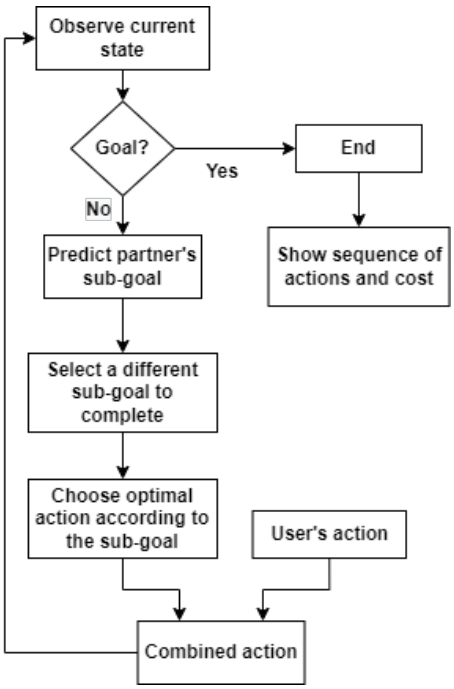

Fig. 12. Goal-predicting Agent algorithm's structure

This approach, although more complex than the others, still follows an individual strategy. Like in the replanning agent model, this algorithm adapts to the partner. The difference is it focuses on the sub-goal instead of each action.

5) Best Shared-Reward Agent model: This model introduces the idea of joint planning which considers that there are no individual moves, only a single collective action. The difficulty of this approach is creating a function that correctly evaluates the shared reward of this combined action (human action + AI action). In the AlphaGo algorithm, a trained neural network was in charge of providing this value.

Since creating the function that estimates the shared reward of a joint action proved extremely challenging, we implemented a model with "hardcoded" values to demonstrate the power of this approach (Figure 13).

We consider this model to be the best representation of the collaboration between agents. The hardcoded heuristic values allow us to evaluate the model's performance and define it as the maximum collaboration ceiling that the other models try to achieve.

VI. Results

In this section, we document the experimental results obtained by our models. We describe the set of problems we used and the defined metrics. We then compare the performance of each model and draw some conclusions.

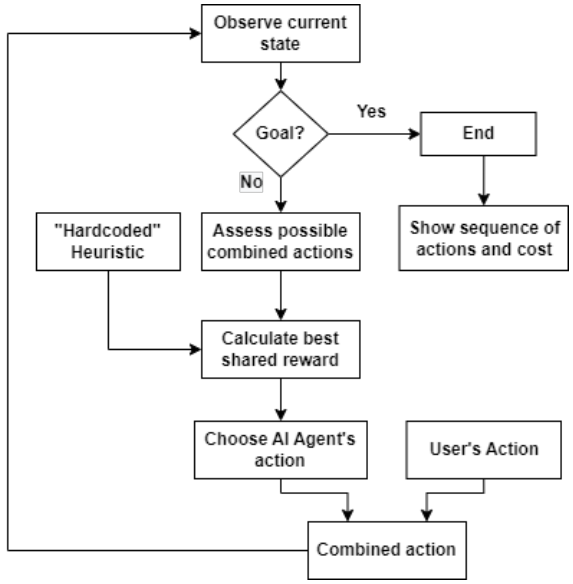

Fig. 13. Best Shared-Reward algorithm's structure

A. Problem set

The performance of our models is evaluated on a randomly generated set of 2D grid search problems. All the items in this set have 5x5 dimensions to ensure the search space doesn't grow too large.

Each problem is a combination of an initial state and a goal state on the grid world. We insert agents and objects in a given position in the grid. The AI agent either works alone (single agent model) or with the help of a human agent. The goal is to pick up all items and deliver them in a given position. The number of items (n) varies between 2 and 7.

B. Performance Metrics

The objective of this thesis is to build an AI Agent model that can collaborate with a human. As previously mentioned, we focus on maze-solving problems. Since we only allow vertical and horizontal moves, there are multiple ways of moving from point A to point B. As the number of actions to achieve the goal state grows, more parallel optimal paths exist.

This fact indicates that it would be **wrong** to measure our model's performance with classical accuracy formulas, such as:

$$Accuracy\ \ (\%) = \frac{correct\ predictions}{total\ predictions} * 100 \qquad (3)$$

Instead, we evaluate the models using a much more suitable metric. If the models correctly identify the sub-goal of the user agent, the AI agent will try completing a different sub-goal. If, on the other hand, the models wrongly identify the user's sub-goal, the AI agent may try to complete that goal itself. This error will translate to an increase in the combined cost of solving the problem.

To give an estimate of the model's performance, we compare the timesteps it takes to solve the problem with that of an ideal

model. In this context, the "ideal model" means that the users adopt a shared strategy (best shared reward model).

We estimate how capable of collaborating a model is by using the expression:

$$C = \frac{t_{ideal\ model}}{t_{current\ model}} \tag{4}$$

where C is the "estimation of collaboration", t_x is the number of timesteps required to find a solution with model x, and the ideal model is the best shared reward model. C has no physical meaning, but it allows us to compare different models.

The second metric we used is the elapsed time to find the solution, t. Here we only consider the time the algorithm spends searching for the AI agent's action. With this we intend to get a view of the computational cost of each model.

We present the experimental results in tables I and II, where n is the number of items in the problem.

Firstly, we can observe that both the single-agent and simple multi-agent models get considerably worst value of C than models that adapt their strategy considering the partner. Furthermore, we can also notice that the second model, although having multiple agents, doesn't collaborate.

The replanning agent results are better than initially expected. This model achieves relatively high values of C, even when n increases. We presume its success comes from the existence of several sub-goals per problem. We mean that since there are many tasks that we can complete concurrently if the algorithm makes a wrong prediction of the partner's action, it doesn't always translate into an increase in cost in the final solution. If we were to use a larger grid with fewer items on it, we expect the algorithm to be heavily penalized for the wrong action predictions. Additionally, we see that the collaboration achieved by this model comes with a relatively high computational cost.

The goal-predicting agent shows the best performance as expected. Just like the replanning agent, it re-evaluates the state of the environment at each time step, achieving high indices of collaboration. However, we demonstrate that predicting the partner's goal can be more efficient than precisely anticipating its next move. We speculate that this increase in performance comes from us defining the set of sub-goals before the agents start acting. Being true, this shows the need for an effective partial-order planner to decompose the goal into smaller task.

VII. Conclusions and Future Work

The Human-robot collaboration field is gaining more relevance due to its increasing number of applications, which emphasizes the need for proper collaborative systems.

We started this work with an introduction to the theoretical concepts that define cooperation between humans. Collaboration requires agents to have an equal representation of the environment, so they can anticipate each other's actions and correctly adapt to them.

We showed that mental states (such as goals) guide human action, with the Theory of Mind being the mechanism that allows humans to comprehend mental states. We also discussed the implementation of this mechanism in robots.

Having highlighted the importance of action prediction, we introduced two approaches for action prediction.

Neural networks are algorithms that use data about the problem to learn patterns.

The second approach we introduced was planning. This methodology uses the structure of the environment (rules and constraints) to choose the sequence of actions that take us closer to a given goal. Here, we also described some algorithms used in classical planning problems. Finally, we introduced a combined application of planning and neural networks and discussed their importance.

The main contribution of this thesis is a new approach to human-robot collaboration. Our model-based algorithm utilizes planning to assess possible actions. We choose the best action with the help of a heuristic function. Ideally, the heuristic function should be substituted for a policy network (reinforcement learning) that evaluates the shared reward of joint actions (as shown in figure 10).

The model this thesis proposes can be expanded in two ways:

- Implement the policy network to estimate heuristic values. We propose that the algorithm uses the Monte Carlo method to train itself. This method can create a joint distribution for the actions of both agents and assess which combination is the most beneficial to the group. The results of this thesis show that goal prediction leads to better cooperation than action prediction. With this in mind, we also propose training a second policy network, where the Monte Carlo method would create a joint distribution for the sub-goals of the agents.
- Incorporate a partial-order planner. This thesis assumes that the problem's goal is always pre-decomposed into sub-goals. This principle might work with objectives that resemble recipes, where there is a known list of tasks to complete. However, in real-world applications, that is not always the case. We need to make our algorithm understand the goal composition to make it collaborate with humans in a new environment. We suggest the development of a partial-order planner to solve the issues of adapting the model to different problems.

References

[1] S. Aoki, C.-W. Lin, and R. Rajkumar, "Human-robot cooperation for autonomous vehicles and human drivers: Challenges and solutions," *IEEE communications magazine*, vol. 59, no. 8, pp. 35–41, 2021.

[2] J. Holland, L. Kingston, C. McCarthy, E. Armstrong, P. O'Dwyer, F. Merz, and M. McConnell, "Service robots in the healthcare sector," *Robotics*, vol. 10, no. 1, p. 47, 2021.

[3] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human–robot collaboration safety in manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 260–275, 2015.

[4] R. Inam, K. Raizer, A. Hata, R. Souza, E. Forsman, E. Cao, and S. Wang, "Risk assessment for human-robot collaboration in an automated warehouse scenario," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 743–751, IEEE, 2018.

TABLE I
VALUES OF C FOR EACH MODEL IN EACH PROBLEM

	Single Agent	Simple Multi-Agent	Replanning Agent	Goal-Predicting Agent
n=2	0,6	0,67	1	1
n=3	0,62	0,68	0,87	1
n=4	0,51	0,53	0,83	0,83
n=5	0,44	0,48	0,85	1
n=6	0,5	0,53	0,86	0,97
n=7	0,49	0,62	1	1
n=8	0,57	0,70	1	1

TABLE II
VALUES OF t FOR EACH MODEL IN EACH PROBLEM

	Single Agent	Simple Multi-Agent	Replanning Agent	Goal-Predicting Agent
n=2	0,06s	0,06s	0,63s	0,07s
n=3	0,01s	0,01s	0,23s	0,13s
n=4	0,10s	0,10s	1,54s	0,40s
n=5	0,07s	0,07s	1,67s	0,6s
n=6	0,10s	0,11s	3,51s	1,33s
n=7	0,10s	0,11s	3,52s	1,34s
n=8	0,06s	0,08s	2,23s	1,07s

[5] D. Archer and R. M. Akert, "Words and everything else: Verbal and nonverbal cues in social interpretation.," *Journal of personality and social psychology*, vol. 35, no. 6, p. 443, 1977.

[6] C. A. Brownell, "Early developments in joint action," *Review of Philosophy and Psychology*, vol. 2, pp. 193–211, Jun 2011.

[7] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: bodies and minds moving together," *Trends in Cognitive Sciences*, vol. 10, no. 2, pp. 70–76, 2006.

[8] D. C. Dennett, "Intentional systems in cognitive ethology: The "panglossian paradigm" defended," *Behavioral and Brain Sciences*, vol. 6, no. 3, pp. 343–355, 1983.

[9] S. Baron-Cohen, "Precursors to a theory of mind: Understanding attention in others," *Natural theories of mind: Evolution, development and simulation of everyday mindreading*, vol. 1, pp. 233–251, 1991.

[10] A. M. Leslie, O. Friedman, and T. P. German, "Core mechanisms in theory of mind," *Trends in cognitive sciences*, vol. 8, no. 12, pp. 528–533, 2004.

[11] R. Brooks, A. Smith, and B. Scassellati, "Foundations for a theory of mind for a humanoid robot," 07 2001.

[12] A. M. Leslie, "Pretense and representation: The origins of" theory of mind."," *Psychological review*, vol. 94, no. 4, p. 412, 1987.

[13] K. A. Loveland and S. H. Landry, "Joint attention and language in autism and developmental language delay," *J Autism Dev Disord*, vol. 16, pp. 335–349, Sept. 1986.

[14] T. Vierkant, "What metarepresentation is for," *Foundations of metacognition*, pp. 279–288, 2012.

[15] M. S. Gobel, H. S. Kim, and D. C. Richardson, "The dual function of social gaze," *Cognition*, vol. 136, pp. 359–364, 2015.

[16] M. Argyle, R. Ingham, F. Alkema, and M. McCallin, "The different functions of gaze," 1973.

[17] C.-M. Huang, S. Andrist, A. Sauppé, and B. Mutlu, "Using gaze patterns to predict task intent in collaboration," *Frontiers in psychology*, vol. 6, p. 1049, 2015.

[18] B. Scassellati, "Theory of mind for a humanoid robot," *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.

[19] S. Baron-Cohen, *Mindblindness: An essay on autism and theory of mind*. MIT press, 1997.

[20] S. Baron-Cohen, "The eye direction detector (edd) and the shared attention mechanism (sam): Two cases for evolutionary psychology.," Lawrence Erlbaum Associates, Inc, 1995.

[21] K. A. McCabe, V. L. Smith, and M. LePore, "Intentionality detection and "mindreading": Why does game form matter?," *Proceedings of the National Academy of Sciences*, vol. 97, no. 8, pp. 4404–4409, 2000.

[22] O. E. David, N. S. Netanyahu, and L. Wolf, "Deepchess: End-to-end deep neural network for automatic learning in chess," in *International Conference on Artificial Neural Networks*, pp. 88–96, Springer, 2016.

[23] G. Haworth and M. Velliste, "Chess endgames and neural networks," *ICGA Journal*, vol. 21, no. 4, pp. 211–227, 1998.

[24] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *International Journal of Computer Vision*, vol. 130, no. 5, pp. 1366–1401, 2022.

[25] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[26] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1080–1088, 2015.

[27] Y. Kong, Z. Tao, and Y. Fu, "Deep sequential context networks for action prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1473–1481, 2017.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] S. Saxena, "Introduction to Long Short Term Memory (LSTM)." https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/, 2021.

[30] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," *Advances in neural information processing systems*, vol. 21, 2008.

[31] S. Şeker, E. Ayaz, and E. Türkcan, "Elman's recurrent neural network applications to condition monitoring in nuclear power plant and rotating machinery," *Engineering applications of artificial intelligence*, vol. 16, no. 7-8, pp. 647–656, 2003.

[32] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Pedestrian action anticipation using contextual feature fusion in stacked rnns," *arXiv preprint arXiv:2005.06582*, 2020.

[33] O. Olabiyi, E. Martinson, V. Chintalapudi, and R. Guo, "Driver action prediction using deep (bidirectional) recurrent neural network," *arXiv preprint arXiv:1706.02257*, 2017.

[34] "What is ai planning?." https://planning.wiki/guide/whatis/aip. Accessed: 2022-09-3.

[35] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[36] B. J. Grosz, "Collaborative systems (aaai-94 presidential address)," *AI Magazine*, vol. 17, p. 67, Mar. 1996.