A Review on Learning Planning Action Models for Socio-Communicative HRI

Ankuj Arora, Humbert Fiorino, Damien Pellier, Sylvie Pesty

A Review on Learning Planning Action Models for Socio-Communicative HRI

Ankuj Arora, Humbert
Fiorino, Damien Pellier
and Sylvie Pesty
Laboratoire LIG, Université
Grenoble-Alpes, Grenoble,
France
firstname.lastname@imag.fr

ABSTRACT

For social robots to be brought more into widespread use in the fields of companionship, care taking and domestic help, they must be capable of demonstrating social intelligence. In order to be acceptable, they must exhibit socio-communicative skills. Classic approaches to program HRI from observed human-human interactions fails to capture the subtlety of multimodal interactions as well as the key structural differences between robots and humans. The former arises due to a difficulty in quantifying and coding multimodal behaviours, while the latter due to a difference of the degrees of liberty between a robot and a human. However, the notion of reverse engineering from multimodal HRI traces to learn the underlying behavioral blueprint of the robot given multimodal traces seems an option worth exploring. With this spirit, the entire HRI can be seen as a sequence of exchanges of speech acts between the robot and human, each act treated as an action, bearing in mind that the entire sequence is goal-driven. Thus, this entire interaction can be treated as a sequence of actions propelling the interaction from its initial to goal state, also known as a plan in the domain of AI planning. In the same domain, this action sequence that stems from plan execution can be represented as a trace. AI techniques, such as machine learning, can be used to learn behavioral models (also known as symbolic action models in AI), intended to be reusable for AI planning, from the aforementioned multimodal traces. This article reviews recent machine learning techniques for learning planning action models which can be applied to the field of HRI with the intent of rendering robots as socio-communicative.

Keywords

AI Planning, Action Models, Machine Learning, Human Robot Interaction, Social Intelligence

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

1. INTRODUCTION

With the near simultaneous advances in mechatronics on the engineering side and ergonomics on the human factors side, the field of social robotics has seen a significant spike in interest in the recent years. Driven with the objective of rendering robots as socio-communicative, there has been an equally heightened interest towards researching techniques to endow robots with cognitive, emotional and social skills. The strategy to do so draws inspiration from study of human behaviors. For robots, social and emotive qualities not only lubricate the interface between humans and robots, but also promote learning, decision making and so on. These qualities strengthen the possibility of acceptability and emotional attachment to the robot [5, 6]. This acceptance is only likely if the robot fulfils a fundamental expectation that one being has of the other: not only to do the right thing, but also at the right time and in the right manner [5]. This social intelligence or 'commonsense' of the robot is what eventually determines its social acceptability in the long run.

Commonsense, however, is not that common. Robots can, thus, only learn to be acceptable with experience. However, teaching a humanoid the subtleties of a social interaction is not evident. Even a standard dialogue exchange integrates the widest possible panel of signs which intervene in the communication and are difficult to codify (synchronization between the expression of the body, the face, the tone of the voice, etc.). In such a scenario, learning the behavioral model of the robot is a promising approach.

Thus, another way of solving this problem is given a set of HRI traces, to learn the interaction script or the behavioral model of the robot which governs this interaction. This learning can be conducted with the help of some fairly recent and other ongoing advances in the field of AI. In the field of AI planning for instance, this entire interaction can be viewed as a series of actions (where each speech act is treated as an action) which take the system from the initial to the goal state, the goal being the successful termination of the interaction [18]. In the literature, AI (or Automated) planning has been used in HRI for robot action planning and reasoning [1] . There has been little work done in using AI planning approaches to empower robots with socio-communicative abilities.

In such domains, planners now leverage recent advancements in machine learning (ML) to recreate the blueprint of the actions applicable to the domain, but cannot be easily identified or programmed, alongwith their signatures, preconditions and effects. Several classical and recent ML techniques can be leveraged to reproduce the underlying behavioral model. This model, once learnt, can further render the humanoid as autonomous and pioneer future HRI interactions. This is a conscious and directed effort to decrease laborious manual coding and increase quality. This article briefly reviews recent machine learning techniques for learning planning action models for the field of HRI.

This article is organized as follows: we start by briefly introducing and explaining the interplay between Automated Planning (AP), Machine Learning (ML) and HRI to solve a common problem. We then represent a classification of various techniques of learning action models, citing examples of each. These examples are then detailed in the following section. We then briefly discusses the persisting issues in the field despite the advances, and terminate with a conclusion.

2. PROBLEM STATEMENT

Consider the case where Automated Planning (AP) were to be used to construct the core behavioral model of a robot to govern its multimodal interactions with humans. It is very difficult, if not impossible, to fine tune such a model by a domain expert to inculcate and account for subtle human behaviours which arise and interplay even in the simplest dialog exchange. However, using ML techniques, it is possible to learn this model from an actual HRI. As demonstrated in the figure 1, the HRI can be viewed as a planning problem: in an initial run of the interaction, the robot speech acts are governed by observation, imitation or demonstration techniques [2, 21]. One particular approach that seems promising is that of 'beaming' by human pilots [3]. Thanks to this technique, a human operator can perceive, analyze, act and interact with a remote person through robotic embodiment. A human operator will solve both the scaling and social problems by optimally exploiting the robots affordances. The following speech act exchange sequence between the robot and human is treated as a sequence of actions which constitutes the trace set (also called the execution experience). This speech act exchange is what drives the interaction from its initial to goal state. An initial state or a goal is composed of a set of predicates. A predicate is a set of constant symbols or variable symbols. It is a set of constant symbols in case it is grounded, in which case it evaluates to true or false. Thus, a single trace is constituted of: initial state predicates, speech act sequence, and the final state predicates.

These traces are then fed to the learner (see figure 2), whose role is to learn the behavioral (action) model m that serves as the 'blueprint' of the actions. An action model is defined by (a, Pre, Add, Del), where a is an action name with zero or more variables (as parameters), Pre, Add and Del being the precondition list, add list and delete list, respectively. The precondition list is the set of conditions (eventually a conjunction of predicates) which need to be satisfied for the action to be triggered in a particular state. The add and delete lists are the set of grounded predicates which will be added or deleted respectively from the current state, upon the application of the action to the current state. This action application then produces the next state, which upon successive action applications leads to the goal state. In the field of AI planning, the model is represented in a standard language called the Planning Domain Definition Language (PDDL). It has been the official language for the representation of the problems and solutions in all of the International Planning Competitions (IPC) which have been held 1998 onwards [14].

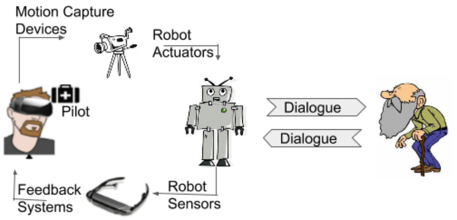


Figure 1: Step 1 - Initial Run of HRI experiment by beaming [3]

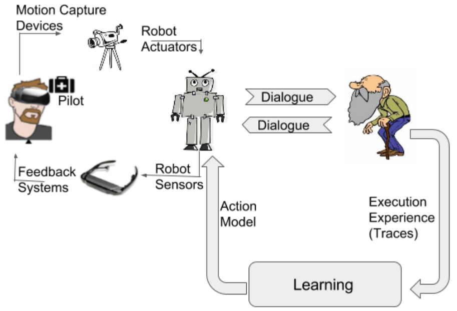
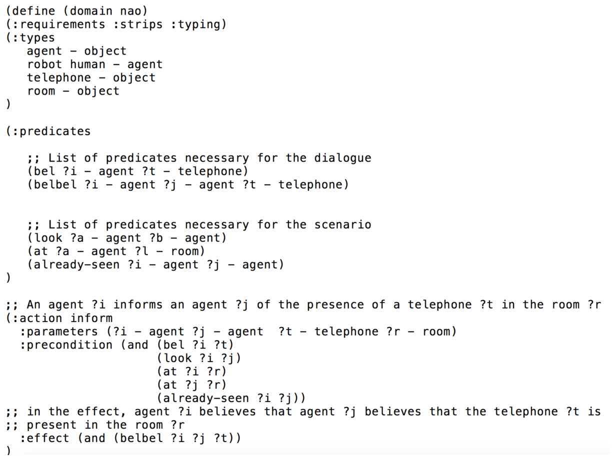


Figure 2: Step 2 - Learning the behavioral model from collected traces



```
(define (domain nao)
(:requirements :strips :typing)
(:types
   agent - object
   robot human - agent
   telephone - object
   room - object
)

(:predicates

   ;; List of predicates necessary for the dialogue
   (bel ?i - agent ?t - telephone)
   (belbel ?i - agent ?j - agent ?t - telephone)


   ;; List of predicates necessary for the scenario
   (look ?a - agent ?b - agent)
   (at ?a - agent ?l - room)
   (already-seen ?i - agent ?j - agent)
)

;; An agent ?i informs an agent ?j of the presence of a telephone ?t in the room ?r
(:action inform
  :parameters (?i - agent ?j - agent  ?t - telephone ?r - room)
  :precondition (and (bel ?i ?t)
                     (look ?i ?j)
                     (at ?i ?r)
                     (at ?j ?r)
                     (already-seen ?i ?j))
;; in the effect, agent ?i believes that agent ?j believes that the telephone ?t is
;; present in the room ?r
  :effect (and (belbel ?i ?j ?t))
)
```

Figure 3: Domain Description and Schema for Operator 'inform' in an HRI domain

A sample action called 'inform' in the PDDL language is represented in the figure 3. The objective of this action is for the agent (in this case the robot) to inform a human about the presence of a telephone in the room. The preconditions for this action: both the robot and human are in the room, the robot believes in the presence of a telephone in the room, and that the robot has seen the human; are represented in the form of predicates. As an effect of this action, the robot believes that the human believes in the presence of a telephone in the room, signifying its successful execution of the action 'inform'.

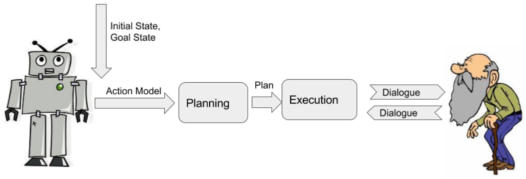


Figure 4: Step 3 - Autonomous re-run of HRI

The challenge lies scripting this action model with more complex speech acts, relying solely on the expertise of a domain expert who also in his own right, is likely to commit errors while scripting.

The effort required by the domain expert to script this subtle and delicate humanoid behavioral model can be diminished by ML. The work done in ML goes hand in hand with the long history of planning, as ML is viewed as a potentially powerful means of endowing an agent with greater autonomy and flexibility. It often compensates for the designer's incomplete knowledge of the world that the agent will face. The developed ML techniques could be applied across a wide variety of domains to speed up planning. This is done by learning the underlying behavioral model from the experience accumulated during the planning and execution phases (refers to speech act exchanges). These employed learning techniques vary widely in terms of context of application, technique of application, adopted learning methodology and information learned.

Once the model m is learnt, it is fed to a planner (see figure 4) along with an initial state $s0$ and goal state g. Together, all three constitute a planning problem which is defined by $(s0, g, m)$. A solution to a planning problem is a plan composed of an action sequence $(a1, a2, an)$, where the actions guide the transition of the system from the initial to the goal state [28]. Thus, the learnt model can be re-usable to plan future dialogue sequences between the robot and the human, in such a way that the need of a 'teacher' to govern the robot behavior is suppressed, and the robot can interact autonomously.

In summary, Machine Learning (ML) is increasingly being used to resolve the aforementioned planning problem. This article tries to classify various approaches based on several criterion.

3. LEARNING PLANNING ACTION MODELS

The techniques for learning planning action models can be classified as depicted in figure 5.

3.1 State Observability and Action Effects

The determination of the current state of the system after the action execution may be flawed because of a faulty sensor calibration. Thus, in the case of partial observability of a system, it may be assumed to be in one of a set of 'belief states'.

Similarly action effects may be probabilistic, which means that in a real world scenario, it is not necessary that a unitary action be applicable to a state. On the contrary, multiple actions may be applied, each with a different execution probability.

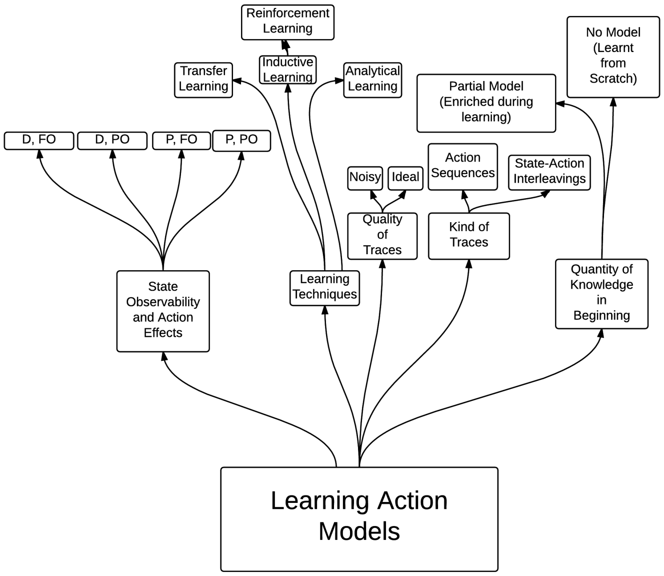


Figure 5: Learning Planning Action Models (D=Deterministic, P=Probabilistic, FO=Fully Observable, PO=Partially Observable)

Keeping these variations of action effects and state observability in mind, we define four categories of implementations:

- Deterministic effects, full state observability: For example, the EXPO [23, 12, 9] system.
- Deterministic effects, partial state observability: In this family, the system may be in one of a set of 'belief states' after the execution of each action. For example, ARMS (Action-Relation Modelling System) [25].
- Probabilistic effects, full state observability: For example, PELA (Planning, Execution and Learning Architecture)[11].
- Probabilistic Effects, Partial State Observability: Barring a few initial works in this area, this classification remains as the most understudied one to date, with no general approach in sight either (for example, Yoon et al. [26]).

3.2 Learning Techniques

This section introduces some classic as well as recently prominent learning techniques that have been successfully used in learning action models. The following subsections have not been conceptualized as learning families, but as orthogonal (and sometimes overlapping) techniques.

3.2.1 Inductive and Analytical Learning

- Inductive learning: The learning system is confronted with a hypothesis space H and a set of training examples D. The desired output is a hypothesis h from H that is consistent with these training examples. Inductive methods generate statistically justified hypotheses [32]. The heart of the learning problem is generalizing successfully from examples. In these cases, inductive techniques that can identify patterns over many examples in the absence of a domain model can come in handy. One prominent inductive learning technique is that of decision tree and regression tree learning. Regression trees offer the advantages of being able to predict a continuous variable and the ability to model noise in the data. A regression tree predicts a value along the dependent dimension for all environmental observations, in contrast to a decision tree, which enables a prediction along a categorical variable (i.e., class).

- Analytic learning: The learning system is confronted with the same hypothesis space and training examples as for inductive learning. However, the learner has an additional input: background knowledge B that can explain observed training examples. The desired output is a hypothesis h from H that is consistent with both the training examples D and the background knowledge B [32]. Analytic learning leans on the learner's background knowledge to analyze a given training instance to identify the relevant features.

More details about classical techniques which have been comprehensively used in operator learning can be found in [32]. The current article sheds light on certain interesting techniques which have more recently come to light and offer interesting possibilities with respect to the task at hand, which is that of learning operators.

3.2.2 Transfer Learning

Many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data. In many real world applications, it is expensive or impossible to re-collect the needed training data and rebuild the models. It would be nice to reduce the need and effort to recollect the training data. In such cases, knowledge transfer or transfer learning between task domains would be desirable. Transfer learning [16], in contrast, allows the domains, tasks, and distributions used in training and testing to be different. In the real world, we observe many examples of transfer learning. For example, we may find that learning to recognize apples might help to recognize oranges. Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task when the latter has fewer high-quality training data [16].

The advantages of using transfer learning are centered around the fact that a change of features, domains, tasks, and distributions from the training to the testing phase does not require the statistical model to be rebuilt. The disadvantages, however, are listed as follows:

- Many proposed transfer learning algorithms assume that the source and target domains are related to each other in some sense. However, if the assumption does not hold, negative transfer may happen, which is worse than no transfer at all (for example, an American tourist learning to drive on the left side of the road in the UK for the first time). In order to avoid negative transfer learning, we need to first study transferability between source domains and target domains. Based on suitable transferability measures, we can then select relevant source domains/tasks to extract knowledge for learning the target tasks.
- Most existing transfer learning algorithms so far have focused on improving generalization across different distributions between source and target domains or tasks. In doing so, they assumed that the feature spaces between the source and target domains are the same. However, in many applications, we may wish to transfer knowledge across domains or tasks that have different feature spaces, and transfer from multiple such source domains. This type of transfer learning is referred to as heterogeneous transfer learning, which is a persisting challenge.
- Has mainly been applied to small scale applications [16].

One particular implementation is an algorithm called LAWS (Learn Action models with transferring knowledge from a related source domain via Web Search) [31].

3.2.3 Reinforcement Learning

Reinforcement learning (RL) [4] is a specific case of inductive learning, and defined more clearly by characterizing a learning problem instead of a learning technique. A general reinforcement learning problem can be seen as composed of just three elements: (1) goals an agent must achieve, (2) an observable environment, and (3) actions an agent can take to affect the environment. Through trial-and-error online visitation of states in its environment, such a reinforcement learning system seeks to find an optimal policy for achieving the problem goals. The strength of reinforcement learning lies in its ability to handle stochastic environments in which the domain theory is either unknown or incomplete. With respect to the planning-learning goal dimension, reinforcement learning can be viewed as both 'improving plan quality' (the process moves toward the optimal policy) and 'learning the domain theory' (begins without a model of transition probability between states) [32]. However, one of the major drawbacks of RL stems from the fact that in its bid to achieve particular goals, it cannot gather general knowledge of the system dynamics, leading to a problem of generalization. RL is particularly interesting for robotics, for this approach often involves learning to achieve particular goals, without gathering any general knowledge of the world dynamics. As a result, the robots can learn to do particular tasks without having trouble generalizing to new ones [17]. For example, LOPE (Learning by Observation in Planning Environments) [8].

3.2.4 Surprise-Based Learning(SBL)

In Surprise-Based Learning (SBL) [20] a surprise is produced if the latest prediction is noticeably different from the latest observation. After performing an action, the world is sensed via the perceptor module which extracts feature information from one or more sensors. If the algorithm had made a prediction, the surprise analyzer will validate it. If the prediction was incorrect, the model modifier will adjust the world model accordingly. Based on the updated model the action selector will perform the next action so as to repeat the learning cycle (see figure 6).

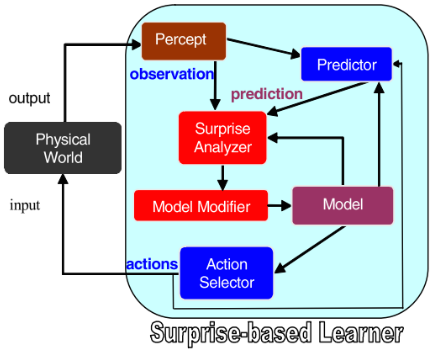


Figure 6: Overview of surprise based learning [20]

A series of approaches based on SBL have used Goal Driven Autonomy (GDA). GDA is a conceptual model for creating an autonomous agent that monitors a set of expectations during plan execution, detects when discrepancies occur, builds explanations for the cause of failures, and formulates new goals to pursue when planning failures arise. In order to identify when planning failures occur, a GDA agent requires the planning component to generate an expectation of world state after executing each action in the execution environment. The GDA model thus provides a framework for creating agents capable of responding to unanticipated failures during plan execution in complex, dynamic environment [24]. For example, the system FOOLMETWICE [15].

3.3 Quality of traces

The execution traces may be classified into pure or adulterated as follows:

- Noisy: The traces can be adulterated because of sensor miscalibration or faulty annotation by a domain expert. For instance, AMAN (Action-Model Acquisition from Noisy plan traces) [28] belongs to this category.

- Ideal: There is no discrepancy between the ideal action and the recorded action. For example, the system OBSERVER [23] falls into this category.

3.4 Kind of traces

This refers to the elements (state information, action information) which constitute the traces. These can be divided into the following:

- Action Sequences: Refers to the case where the executed plan traces can be represented in the form of a sequence of action executions. For example, Opmaker [13].

- State-Action Interleavings: The case where the executed plan traces can be represented in the form of a sequence of alternate state-action representations. For example, LAMP (Learning Action Models from Plan traces) proposed by [30].

3.5 Availability of model in beginning

Before the learning phase begins, the action model may exist in one of the following capacities:

Table 1: Representation Languages

Language	Features	Reference
PDDL (Planning Domain Description Language)	Machine-readable, standardized syntax for representing STRIPS and other languages. Has types, constants, predicates and actions	[14]
STRIPS (Stanford Research Institute Problem Solver)	Sublanguage of PDDL	[7]
OCL (Object Centered Language)	High level language with representation centered around objects instead of states	[13]
STRIPS+WS	STRIPS + functional terms, leading to higher expressiveness	[22]

- No Model: This refers to the fact that the no information on the actions that constitute the model is available in the beginning, and the entire model must be learnt from scratch. For example, OBSERVER [23].

- Partial Model: Some elements of the model are available to the learner in the beginning, and the model is enriched with more knowledge at the end of the learning phase. For example, RIM (Refining Incomplete planning domain Models through plan traces) [29].

3.6 Representation Language

The ideal language would be able to compactly model every action effect the agent might encounter, and no others. Choosing a good representation language provides a strong bias for any algorithm that will learn models in that language. Some languages and their features are summarized in the table 1.

4. STATE OF THE ART

Brief descriptions of some key algorithms corresponding to the aforementioned classification can be found in the following section. These algorithms are also summarized in the table 2.

4.1 OBSERVER

OBSERVER [23] is a system that learns operator preconditions by creating and updating both most specific representation and a most general representation for the preconditions, based on operator executions while solving practice problems. It also learns operator effects by generalizing the delta-state (the difference between post-state and pre-state) from multiple observations.

4.2 RIM

RIM (Refining Incomplete planning domain Models through plan traces) [29] constructs sets of soft and hard constraints which are solved using a weighted MAX-SAT solver to obtain sets of macro-operators and (refined) action models.

4.3 OpMaker

Opmaker [13] is a mixed initiative (where both the human and the machine take initiative), graphical knowledge acquisition tool for inducing parametrized, hierarchical (each object may have relations and attributes inherited from different levels [13]) operator descriptions from example action sequences and declarative domain knowledge, with the minimum of user interaction. It is implemented inside of a graphic tool called GIPO (Graphical Interface for Planning with Objects), which facilitates domain knowledge capture and domain modelling ([13, 10]), a perfect tool for novice users to create plans and learn models with minimum effort.

4.4 LAMP

LAMP (Learning Action Models from Plan traces) [30] learn action models with quantifiers and logical implications. Firstly, the input plan traces (including observed states and actions) are encoded into propositional formulas, which is a conjunction of ground literals to store into a database as a collection of facts. Secondly, candidate formulas are generated according to the predicate lists and domain constraints. Thirdly, a Markov Logic Network (MLN) uses the formulas generated in the above two steps to select the most likely subset from the set of candidate formulas. Finally, this subset is converted into the final action models.

4.5 AMAN

AMAN (Action-Model Acquisition from Noisy plan traces) [28] finds a domain model that best explains the observed noisy plan traces. First, a set of candidate domain models is built by scanning each action in plan traces and substituting its instantiated parameters with their corresponding variables. This is followed by a graphical model to capture the relationship between the current state, correct action, observed action and the domain model. Afterwards the parameters of the graphical model are learnt, following which AMAN generates a set of action models according to the learnt parameters.

4.6 EXPO

The EXPO ([23, 12, 9]) system refines incomplete planning operators by ORM (operator refinement method). EXPO does this by generating plans and monitoring their execution to detect the differences between the state predicted according to the internal action model and the observed state. EXPO then constructs a set of specific hypotheses to fix the detected differences. After being heuristically filtered, each hypothesis is tested in turn with an experiment and a plan is constructed to achieve the situation required to carry out the experiment.

4.7 ARMS

The ARMS (Action-Relation Modelling System) [25] system learns an action model in two phases. In phase one of the algorithm, ARMS finds frequent action sets from plans that share a common set of parameters. In addition, ARMS finds some frequent relation-action pairs with the help of the initial state and the goal state. These relation-action pairs give us an initial guess on the preconditions, add lists and delete lists of actions in this subset. These action subsets and pairs are used to obtain a set of constraints that must hold in order to make the plans correct. The constraints extracted from the plans are then transformed into a weighted MAX-SAT representation, the solution to which produces action models. The process iterates until all actions are modelled.

4.8 PELA

PELA (Planning, Execution and Learning Architecture) [11] performs the three functions suggested in its name. The learning component allows PELA to generate probabilistic rules about the execution of actions. PELA generates these rules from the execution of plans and compiles them to upgrade its deterministic planning model. This is done by performing multiclass classification, which further consists of finding the smallest decision tree that fits a given data set following a Top-Down Induction of Decision Trees (TDIDT) algorithm [19].

4.9 LAWS

LAWS (Learn Action models with transferring knowledge from a related source domain via Web search) [31] makes use of action-models already created beforehand in other related domains, which are called source domains, to help learn actions in a target domain. The target domain and a related source domain are bridged by searching Web pages related to the target domain and the source domain, and then building a mapping between them by means of a similarity function done by calculating the similarity between their corresponding Web pages. The similarity is calculated using the Kullback-Leibler (KL) divergence. Based on the calculated similarity, a set of weighted constraints, called web constraints, are built. Based any available example plan traces in the target domain, other constraints such as state constraints, action constraints and plan constraints, are also built. All the above constraints are solved using a weighted MAX-SAT solver, and target-domain action models are generated based on the solution to the constraint satisfaction problem.

4.10 LOPE

LOPE (Learning by Observation in Planning Environments) [8] learns by sharing among multi agent systems. Learning is performed by three integrated techniques: rote learning of an experience (observation) by creating an operator directly from it, heuristic generalization of incorrect learned operators; and a global reinforcement strategy of operators by rewarding and punishing them based on their success in predicting the behavior of the environment. Reinforcement of an operator means punishment of similar ones, so there is a global reinforcement of the same action. This global reinforcement is done by means of a virtual generalized Q table [8].

4.11 FOOLMETWICE

FOOLMETWICE [15] is a goal-oriented (GDA-Goal Driven Agent) algorithm which learns from surprises. It tries to find inaccuracies in environment model M by attempting to explain all observations received. When a consistent explanation cannot be found, it infers that some unknown event E happened that is not represented in M. After determining when unknown events occur, it creates a model of their preconditions requires generalizing over the states that trigger them.

5. OPEN ISSUES

Table 2: State Of The Art Planning Algorithms (D=Deterministic, P=Probabilistic, FO=Fully Observable, PO=Partially Observable, PDL=Prodigy Description Language)

Algorithm/ Author	Input	Output	Language	Technique	Merits	Demerits	Envi ro nment	Robust to Noise?
OBSE RVER ([23])	Traces, practice problems and description language	Operators	STRIPS-like	Conservative specific-to-general inductive generalization process	(i) Can find out negated preconditions, conditional preconditions and conditional effects (ii) Does not require strong background knowledge	Domain knowledge can be incomplete or incorrect in the following ways : over-general preconditions, over-specific preconditions, incomplete effects, extraneous effects, and missing operators	D, FO	N
RIM ([29])	Incomplete action models and plan traces	Macro-operators and action models	STRIPS	MAX-SAT	Increases accuracy of incomplete operators	Cannot handle incorrect partial initial models	D, FO	Y
Opmaker ([13])	Partial Model, Action sequences	Operator Schema	OCL	Operator induction by mixed initiative	(i) Eases task of operator encoding, fits well into engineering environment for planning domain acquisition and modeling (ii) Useful for non-experts	Needs user input for intermediate state information	D, FO	Y
LAMP ([30])	State-Action Interleavings	Action Models	PDDL	Markov Logic Network (MLN)	More expressive models-inculcate quantifiers and logical implications	Looses efficiency with increasing domain size	D, FO	N
AMAN ([28])	Action sequence	Operators	STRIPS	Gradient Descent, Reinforcement Learning	No background knowledge needed	Model sampling mechanism unclear	D, FO	Y
EXPO ([23])	Incomplete operator set, traces of state sequences	New preconditions, effects, conditional effects, operators, attribute values	PDL	Learning-by-experimentation for Operator Refinement	(i) Learns conditional effects (ii) Methods are goal-directed and learning is incremental	Rules learnt from general to specific	D, FO	N
ARMS ([25])	Action sequence with partial traces	Action Models	STRIPS	Builds a weighted propositional satisfiability problem and solves it using weighted MAX-SAT solver	Can handle cases when intermediate state observations are difficult to acquire	(i) Cannot learn action models with quantifiers or implications (ii) Cannot learn complex action models	D, PO	N
PELA ([11])	Planning problem, STRIPS action model	Enriched action model with planning possibilities in probabilistic domains	PDDL	TDIDT (Top Down Induction of Decision Trees)	Based on off-the-shelf planning and learning components	Assumes initial action model of environment	P, FO	N
LAWS ([31])	Action schemas, predicates, plan traces from target domain, action models from source domain	Action models in target domain	STRIPS	Transfer Learning ,KL divergence	Web exploited as knowledge source	Negative Transfer: When source domain and target domain are not related to each other, brute-force transfer may be unsuccessful	D, FO	N
LOPE ([8])	Number of execution cycles, possible action set	Operators	Propos itional Logic	Reinforcement Learning	Allows knowledge sharing among agents, increasing percentage of successful plans	Differences in sensors of agents causes different ways of perceiving the world, and, therefore, different biases towards operator generation	D, FO	N
FOOL ME TWICE ([15])	Initial state, goal state, model	Expectation, discrepancies, goals	PDDL+	Goal-oriented model based on surprise and explanation generation	Goal-oriented rather than reward-driven, thus allowing frequent goal change without requiring substantial policy re-learning	Cannot acquire exogenous event models with continuous conditions	D, PO	N

Despite the bright prospects that the aforementioned approaches offer, there persist some open issues and loopholes which are discussed as follows:

- Learning with the time dimension: Time plays an imperative role most real life domains. For example, each dialogue in a HRI is composed of an utterance further accompanied by gestural, body and eye movements, all of them interleaved in a narrow time frame. These interactions may thus be represented by a time sequence, with the intent of learning the underlying action model. Barring some initial works in this area, time remains an interesting aspect to explore [27].

- Direct re-applicability of learned model for dialogue exchange: re-use of a learned model by a planner continues to remain a concern. A model that has been learned by applying ML techniques is more often than not incomplete, or more concretely inadept to be fed to a planner to directly generate plans, or in the case of HRI, to reproduce a multimodal dialogue which respects social rules. It needs to be retouched and fine tuned by a domain expert in order to be reusable. This marks a stark incapability of the prominently used machine learning techniques to be and comprehensive, and leaves scope for much more research.

- Extension of classical planning to a full scope domain: The applicability of the aforementioned approaches, most of which have been tested on highly simplified toy domains and not in real scenarios, remains an issue to be addressed. Classical planning refers to a world model in which predicates are propositional: they do not change unless the planning agent acts to change them, all relevant attributes can be observed at any time, the impact of executing an action on the environment is known and deterministic, the effects of taking an action occurring instantly and so on. However, the real world is laced with unpredictability: a predicate might switch its value spontaneously, the world may have hidden variables, the exact impact of actions may be unpredictable, the actions may have durations and so on [32].

6. CONCLUSION

This article argues for the usage of AI planning techniques with the intent of endowing robots with socio-communicative skills, thus augmenting their acceptability. It justifies the notion of learning the underlying behavioral blueprint of the robot from a set of multimodal HRI traces. This learning is achieved by the usage of several state-of-the-art and classical Machine Learning (ML) techniques. The article tries to classify various ML approaches based on several criterion and conditions, along with the merits and demerits of each approach. It then broadly highlights some persisting open issues with the discussed approaches, concluding that a significant number of prominent and interesting techniques have been applied to highly controlled experimental setups, and their application to a real world HRI scenario is a topic of further research.

7. REFERENCES

[1] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila. Task planning for human-robot interaction. In *Proceedings of the joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 81–85. ACM, 2005.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] G. Bailly, F. Elisei, and M. Sauze. Beaming the gaze of a humanoid robot. In *Human-Robot Interaction*, 2015.

[4] A. G. Barto. Reinforcement learning and adaptive critic methods. *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*, pages 469–491, 1992.

[5] C. Breazeal. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59(1):119–155, 2003.

[6] C. Breazeal. Social interactions in hri: the robot view. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):181–186, 2004.

[7] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[8] R. García-Martínez and D. Borrajo. An integrated approach of learning, planning, and execution. *Journal of Intelligent and Robotic Systems*, 29(1):47–78, 2000.

[9] Y. Gil. Acquiring domain knowledge for planning by experimentation. Technical report, DTIC Document, 1992.

[10] R. Jilani, A. Crampton, D. E. Kitchin, and M. Vallati. Automated knowledge engineering tools in planning: state-of-the-art and future challenges. 2014.

[11] S. Jiménez, F. Fernández, and D. Borrajo. The PELA architecture: integrating planning and learning to improve execution. In *National Conference on Artificial Intelligence*, pages 1294–1299, 2008.

[12] S. Jiménez, F. Fernández, and D. Borrajo. Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, 29(1):1–36, 2013.

[13] T. L. McCluskey, N. E. Richardson, and R. M. Simpson. An interactive method for inducing operator descriptions. In *Artificial Intelligence Planning Systems*, pages 121–130, 2002.

[14] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[15] M. Molineaux and D. W. Aha. Learning unknown event models. Technical report, DTIC Document, 2014.

[16] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[17] H. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning probabilistic relational planning rules. In *International Conference on Automated Planning and Scheduling*, pages 73–82, 2004.

[18] C. R. Perrault and J. F. Allen. A plan-based analysis of indirect speech acts. *Computational Linguistics*, 6(3-4):167–182, 1980.

[19] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[20] N. Ranasinghe and W. Shen. Surprise-based learning for developmental robotics. In *Learning and Adaptive Behaviors for Robotic Systems, 2008. LAB-RS'08. ECSIS Symposium on*, pages 65–70. IEEE, 2008.

[21] N. Verstaevel, C. Régis, M.-P. Gleizes, and F. Robert. Principles and experimentations of self-organizing embedded agents allowing learning from demonstration in ambient robotic. *Procedia Computer Science*, 52:194–201, 2015.

[22] T. J. Walsh and M. L. Littman. Efficient learning of action schemas and web-service descriptions. In *Association for the Advancement of Artificial Intelligence*, pages 714–719, 2008.

[23] X. Wang. *Learning planning operators by observation and practice*. PhD thesis, Carnegie Mellon University, 1996.

[24] B. G. Weber, M. Mateas, and A. Jhala. Learning from demonstration for goal-driven autonomy. In *Association for the Advancement of Artificial Intelligence*, 2012.

[25] Q. Yang, K. Wu, and Y. Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171(2):107–143, 2007.

[26] S. Yoon and S. Kambhampati. Towards model-lite planning: A proposal for learning & planning with incomplete domain models. In *ICAPS Workshop on AI Planning and Learning*, 2007.

[27] Y. Zhang, S. Sreedharan, and S. Kambhampati. Capability models and their applications in planning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1151–1159, 2015.

[28] H. H. Zhuo and S. Kambhampati. Action-model acquisition from noisy plan traces. In *Proceedings of the international joint conference on Artificial Intelligence*, pages 2444–2450, 2013.

[29] H. H. Zhuo, T. Nguyen, and S. Kambhampati. Refining incomplete planning domain models through plan traces. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2451–2457. AAAI Press, 2013.

[30] H. H. Zhuo, Q. Yang, D. H. Hu, and L. Li. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, 174(18):1540–1569, 2010.

[31] H. H. Zhuo, Q. Yang, R. Pan, and L. Li. Cross-domain action-model acquisition for planning via web search. In *International Conference on Automated Planning and Scheduling*, 2011.

[32] T. Zimmerman and S. Kambhampati. Learning-assisted automated planning: looking back, taking stock, going forward. *AI Magazine*, 24(2):73, 2003.