Fast and Slow Goal Recognition

Mattia Chiari
University of Brescia
Brescia, Italy
mattia.chiari@unibs.it

Alfonso Emilio Gerevini
University of Brescia
Brescia, Italy
alfonso.gerevini@unibs.it

Andrea Loreggia
University of Brescia
Brescia, Italy
andrea.loreggia@unibs.it

Luca Putelli
University of Brescia
Brescia, Italy
luca.putelli@unibs.it

Ivan Serina
University of Brescia
Brescia, Italy
ivan.serina@unibs.it

ABSTRACT

Goal recognition is a crucial aspect of understanding the intentions and objectives of agents by observing some of their actions. The most prominent approaches to goal recognition can be divided into two main categories: (1) trustworthy systems, which exploit automated reasoning for computing plans compatible with the observed actions, and (2) swifter systems, which try to quickly infer goals, often overlooking complex cognitive processes, and have no formal guarantees of their results. This paper introduces a novel approach inspired by the dual process theory, which integrates these two techniques. A dual-process model is proposed, leveraging fast, experience-based recognition for immediate goal identification, and slow, deliberate analysis for deeper understanding. Machine learning techniques and classical planning techniques are employed to obtain this dual-process system. Experimental evaluations demonstrate the effectiveness of the approach, reducing the amount of resources required to compute a solution (e.g., time to find a goal), while at the same time enhancing accuracy and robustness, especially in more complex scenarios.

KEYWORDS

Goal Recognition, Fast and Slow, Planning, Deep Learning

ACM Reference Format:
Mattia Chiari, Alfonso Emilio Gerevini, Andrea Loreggia, Luca Putelli, and Ivan Serina. 2024. Fast and Slow Goal Recognition. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024,* IFAAMAS, 9 pages.

1 INTRODUCTION

Goal recognition plays a vital role in understanding the intentions and objectives of agents by observing the behavior of the agent, and has several applications in various domains, ranging from robotics to intelligent systems [16, 37]. Traditional goal recognition approaches can be divided into two categories: The first category is made by planning-based approaches which exploit automated reasoning in a known domain and often provide formal guarantees of the correctness of their results or, at least, their compatibility with the observed behavior [29, 30, 35]. The second is made by simpler approaches which try to swiftly infer the agents' goals, exploiting more intuitive techniques such as landmarks [27, 28] or deep learning [9, 10]. Reasoning methods that prioritize the assurance of correctness give the advantage of producing reliable and verifiable results. However, these methods often suffer from slow execution for large-scale or time-sensitive tasks. On the other hand, machine learning approaches excel in providing fast solutions once trained, making them suitable for quick decision-making. This speed comes at the expense of introducing a level of uncertainty or probability associated with the proposed solutions. Inspired by the "Thinking, Fast and Slow" book [22], which highlights the distinct cognitive processes involved in decision-making, this paper presents a novel approach to goal recognition that takes both of these approaches into account, integrating "intuitive" (experience-guided) decisions and deliberative reasoning. As already suggested in the literature for different scenarios [4, 8, 19], by integrating concepts from cognitive psychology and planning, we propose a dual-process model for goal recognition that accounts for both quick, intuitive recognition and slower, deliberate analysis [13, 14, 32]. The proposed approach leverages the strengths of both processes: fast, intuitive recognition for immediate goal identification and slow, deliberate analysis for deeper understanding and inference. In our approach, we leverage a combination of machine learning techniques and planning-based reasoning to model the dual-process system. To effectively orchestrate this dual process framework, we developed a metacognitive agent, as suggested in the existing literature [15].

The implemented system is called FSGR (Fast and Slow Goal Recognition). FSGR is evaluated by an experimental analysis based on known planning domain benchmarks that compares its performance with other existing goal recognition methods. The results demonstrate that FSGR improves the accuracy and robustness of the goal recognition methods, particularly in scenarios with complex and ambiguous goal structures.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

2 BACKGROUND

In this section, we provide a contextual foundation and an overview of the key concepts that underpin the present study.

2.1 Thinking, Fast and Slow

Advancements in algorithms, techniques, computational power, and specialized hardware have made automated reasoning tools more efficient and reliable. However, they often still lack some desired properties that are common in human intelligence, such as generalizability, robustness, and abstraction. To address these limitations, a growing number of AI experts are striving to develop systems that possess more human-like properties. One of the main strategies is to create cognitive architectures that utilize a combination of neural networks and symbolic/logic-based AI. This paper explores multi-agent planning within the context of one such architecture [8], inspired by Kahneman's "Thinking, Fast and Slow" book [22]. Kahneman adopted the terms System 1 (fast thinking) and System 2 (slow thinking) to describe two processes that handle decision-making in different ways. The terminology was first coined by Stanovich and West [36]. System 1 makes intuitive and imprecise decisions, while System 2 deals with complex, logical, and rational decision-making. The division of responsibilities between the two systems is based on the difficulty of the problem and the experience gained in solving it. Over time, System 2 accumulates examples that System 1 can later use with ease. However, System 1 and System 2 are not multi-agent systems, but rather they encapsulate two broad categories of information processing.

2.2 Goal Recognition

Goal recognition (GR) is defined as the task of identifying the intention (goal) of an agent from observations about the agent's behavior in an environment. These observations can be represented as an ordered sequence of discrete actions (each one possibly identified by activity recognition), while the agent's goal can be expressed either as a set of propositions or a probability distribution over alternative sets of propositions (each even forming a distinct candidate goal).

In the approach to GR known as "goal recognition over a domain theory" [30, 37], the available knowledge consists of an underlying model of the behavior of the agent and of the environment. Such a model represents the agent/environment states and the set of actions A that the agent can take; typically this is specified by a planning language such as PDDL [24]. Given the set of all possible propositions F, also called *fluents* or *facts*, the states of the agent and of the environment are formalized as subsets of F. Each domain action in A is modeled by a set of preconditions and a set of effects, both over F.

An instance of a GR problem $T = \langle \Pi, I, O, \mathcal{G} \rangle$ is specified by: (i) a given domain $\Pi = \langle F, A \rangle$, which specifies the set F of possible fluents and the set of available actions A; (ii) an initial state of the agent and the environment $I \subseteq F$ (iii) a sequence of observations $O = \langle obs_1, .., obs_n \rangle$, with $n \geq 1$, where each $obs_i \in A$ is an action taken by the agent; (iv) a set of possible goals $\mathcal{G} = \{G_1, .., G_m\}$, with $m \geq 1$, where each $G_i \subseteq F$. An observation is a trace of the full sequence of actions π performed by the agent to achieve the goal. This means that an observation is a subsequence of π, whose actions might be non-consecutive but have the same order as in π. Solving a GR instance consists in identifying $G^* \in \mathcal{G}$ that corresponds to the (unknown) goal of the agent. There are two typical approaches for GR: the *model-based goal recognition* (MBGR), in which GR is defined as a reasoning task addressable by automated planning techniques [18, 25], and the *model-free goal recognition* (MFGR) [2, 9, 10, 16], in which GR is formulated as a classification task addressed through machine learning. MFGR requires minimal information about the domain actions (each action is specified by just a label) and it can operate without the specification of an initial state, which can be completely unknown. Moreover, since running a learned classification model is usually fast, an MFGR system is expected to run much faster than an MBGR system based on planning algorithms. On the other hand, MFGR needs a data set of solved GR instances from which to learn a classification model for the new GR instances of the domain.

3 RELATED WORK

Goal recognition has been extensively studied through model-based approaches exploiting planning techniques [25, 28–30, 33, 35], matching techniques relying on plan libraries (e.g., [26]) and more recently with reinforcement learning [2]. The work in [29] computes goal and plan recognition by computing, for all the goals in the hypotheses, an optimal plan from the initial state to the goal and an optimal plan that complies with the observations O from the same initial state to the same goal. The candidate goals are all the goals for which the cost of these two optimal plans is the same. Although this approach is formally exact, its major drawback is that it requires checking all the possible goals in the hypothesis set, with several calls to the planner, which could take a long time. In this work, we adopt this system as the slow thinking component (System 2), comparing our integrated framework to it. The method of Ramirez and Geffner is expanded using standard planners and providing a probability distribution over the candidate goals [30]. Although these works focus on rational agents, the work in [23] focuses on goal recognition with different degrees of rationality of the agents. Besides fully rational agents, another assumption made by most goal recognition approaches is that actions can only be partially observed (meaning that not all actions can be observed), but at least they are observed in the same order they were executed. A study on exploiting off-the-shelf planners without knowing this order is available in [12].

In [27, 28] a faster approach based on landmarks is introduced, called LGR. Unlike [29], for GR instances that have partial observations of the complete plan trace, LGR does not provide any formal guarantee of the correctness of the predicted goal. In this sense, note that systems like LGR could be used in our framework as the fast thinking component (System 1) with a proper configuration of our Metacognitive System. In this case, given that LGR provides a score for each possible goal, our confidence metric, which is explained in Section 4.3, could be exploited in a very similar way. Regarding the experience metric, this would require a more tailored design. However, as shown in [10], when enough training data is provided, our choice for System 1 can achieve more accurate performance than LGR. Moreover, given that our System 1 is a MFGR based on a neural network, it can learn from the problems solved by System 2 and improve over time, providing a better integration with the the slow thinking component which progressively "teaches" System 1. The same improvement cannot be ensured with LGR which is based on planning techniques. Similarly, also the work in [3], which proposes a neuro-symbolic approach that combines learning techniques such as LSTMs (Long Short-Term Memory) and planning techniques to solve goal and plan recognition problems from state traces, could be used as System 1. However, we opted not to use it in this work, as it was tested on a much simpler problems compared to

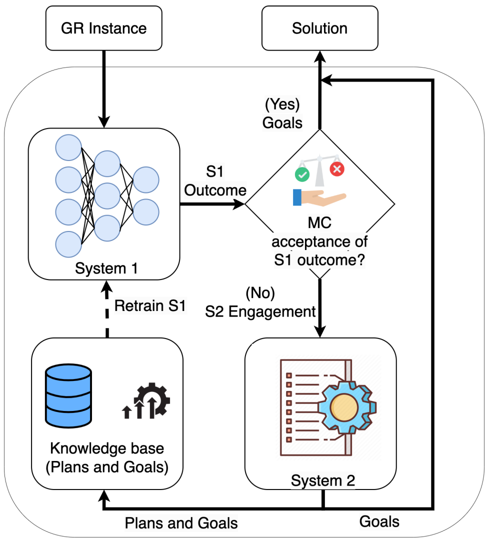

Figure 1: Architecture of FSGR. A GR instance is processed by System 1 (S1) and its solution is evaluated by a metacognitive agent which decides whether to trust it or not. In the latter case, System 2 (S2) is engaged. The plans computed by System 2 are stored in the knowledge base and used for training System 1.

the ones in our dataset (for instance, they consider only 48 fluents for LOGISTICS compared to 154 in our experiments).

4 FAST AND SLOW ARCHITECTURE FOR GOAL RECOGNITION

In this section, we describe the architecture of FSGR. In particular, we show how System 1 (also S1) and System 2 (also S2) are realized and how the metacognitive agent (also MC) orchestrates between the two systems to provide the final prediction. The architecture is depicted in Figure 1. Given a GR instance, System 1 computes a solution based on past experience in the domain. The solution proposed by S1 is then evaluated by the metacognitive agent that decides whether to accept it or engage System 2 for a better solution. In this case, the solution of the instances computed by System 2 are also added to the knowledge base for possible retraining of System 1. We assume that the agent is fully rational and follows optimal plans to achieve its goals, as described in Section 5. Moreover, we assume partial observability (i.e. only a percentage of the actions can be seen) of the agent's actions but no noise in such observations.

4.1 System 1: GRNet

System 1 is developed with an MFGR system based on an LSTM neural network, as proposed in the GRNet approach [10]. The input of the network is a GR instance. The output of GRNet is a score in $[0, 1]$ for each proposition in F. This model can be used for every GR instance over F (the training process is performed once for each domain). The structure of GRNet is shown in Figure 2.

In this system, an embedding layer [6] transforms each action a_i in the input into a vector e_i of real numbers. The index of each action is simply the result of an arbitrary order of actions that is

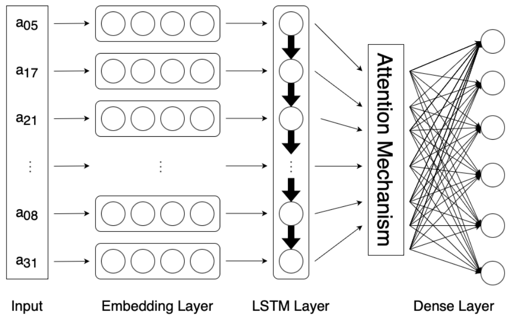

Figure 2: Architecture of System 1. Each action is embedded into a vector and then fed to a LSTM layer and an Attention mechanism.

computed in the pre-processing phase, only once for the domain under consideration. Note that two consecutive observed actions a_i and a_j may not be consecutive in the full plan of the agent, which may contain any number of actions between a_i and a_j.

GRNet is based on a Long Short-Term Memory network (LSTM) [21], which is a kind of neural network specifically suitable for processing sequential data like signals or text documents (in our case the sequence of observed actions) and is composed of several cells, one for each element of the input sequence. The output of each cell is processed by an Attention Mechanism [5], in particular, the variant proposed by Yang et al. [38], which computes the weights representing the contribution of each element of the sequence, and generates a unique representation (also called the *context vector*) of the entire plan trace. The context vector is then passed to a feed-forward layer, which has N output neurons with a *sigmoid* activation function, where N is to the number of domain fluents (propositions) that can appear in any goal of $\mathcal{G}$ for any GR instance in the domain. In our experiments, N was set to the size of the domain fluent set F (i.e., $N = |F|$). The output of the i-th neuron $\bar{o}_i$ corresponds to the i-th fluent f_i (fluents are lexically ordered), and the activation value of $\bar{o}_i$ gives a rank for f_i being true in the agent's goal (with a rank greater than 0.5 meaning that f_i is true in the goal). For each fact f_i, we evaluate network performance using common machine learning metrics, such as precision, recall, or F-Score on a separate validation set, made of instances that the network did not see at training time. In particular, we calculate the precision score as the fraction of goals correctly identified among those predicted by the network as true. We use this score to measure how the neural network performs when it makes a specific prediction. As a loss function, we used the standard binary cross-entropy. The dimension of the embedding vectors, the dimension of the LSTM layer, and other hyper-parameters of the networks are selected using the Bayesian-optimisation approach provided by the Optuna framework [1], with a validation set formed by 20% of the training set, while the remaining 80% is used for training the network.

4.2 System 2: Plan Recognition as Planning

System 2 is based on Plan Recognition as Planning (PRP) [29]. Given an instance of a goal recognition problem $T = \langle \Pi, I, O, \mathcal{G} \rangle$, we want to compute $\mathcal{G}^*_T$, the exact solution to the problem. $\mathcal{G}^*_T$ is a goal set that contains all goals $G \in \mathcal{G}$ such that some optimal plan for

Algorithm 1 MC Algorithm

```
Input:
  • $T = \langle \Pi, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$: the goal recognition instance
  • $y_i$: output of S1 for the current instance
  • $p$: vector of S1 precision scores for each fact on a validation set
  • $\tau_1, \tau_2$: Meta Classifier thresholds
Output: A set of predicted goals
1:  $\hat{G} \leftarrow \arg\max_{G \in \mathcal{G}} mean_{f \in G}(y_i[f])$
2:  $\hat{G}_2 \leftarrow \arg\max_{G \in \mathcal{G} - \{\hat{G}\}} mean_{f \in G}(y_i[f])$
3:  $confidence \leftarrow mean_{f \in \hat{G}}(y_i[f]) - mean_{f \in \hat{G}_2}(y_i[f])$
4:  $count \leftarrow 0$
5:  $sum \leftarrow 0$
6:  for $f \in \hat{G}$ do
7:      if $y_i[f] > 0.5$ then
8:          $sum \leftarrow sum + p[f]$
9:          $count \leftarrow count + 1$
10:     end if
11: end for
12: $experience \leftarrow sum/(count + \epsilon)$
13: if $confidence > \tau_1/|\hat{G}|$ & $experience > \tau_2$ then
14:     return $\{ \hat{G} \}$
15: else if $experience > \tau_2$ then
16:     $goalset \leftarrow \{G \in \mathcal{G} | 0 \leq mean_{f \in \hat{G}} y_i[f] - mean_{f \in G} y_i[f] \leq \tau_1/|\hat{G}|\}$
17: else
18:     $goalset \leftarrow \mathcal{G}$
19: end if
20: $results \leftarrow S2(\langle \Pi, \mathcal{I}, \mathcal{O}, goalset \rangle)$
21: if $results == \{\}$ then
22:     return $\{\hat{G}\}$
23: else
24:     return $result$
25: end if
```

$P = \langle \Pi, I, G \rangle$ satisfies the observation sequence O. To obtain this goal set, we run an optimal planner twice for each goal in $\mathcal{G}$. First, we compute an optimal solution, if it exists, to the planning problem $P_G = \langle \Pi, I, G \rangle$ for all $G \in \mathcal{G}$; then again, for all $G \in \mathcal{G}$, we compute an optimal solution to a transformed planning problem P'_G whose solution is a solution for P_G that satisfies the observations in O. To do that, we followed the work in Ramírez and Geffner [30], where P'_G is defined as a tuple $\langle \Pi', I, G' \rangle$, with $\Pi' = \langle F \cup \{p_a | a \in O\}, A' \rangle$, $G' = \{G \cup \{p_a | a \in O\}\}$, and A' is a new set of actions derived from A as explained below. To each $a \in A$ corresponds an action $a' \in A'$ with the same structure as a; for each action a' that also appears in the observation sequence O, we add an additional effect. This extra effect is set to be p_a when a is the first observation in O, otherwise, it is set to $p_b \implies p_a$ when b is the action that immediately precedes a in O. The candidate goals in $\mathcal{G}^*_T$ are all the goals in G so that the cost of an optimal solution for P_G has the same cost as an optimal solution for P'_G. In our implementation, we compile the problems using PAC-C [7], this is possible by converting P'_G into a PAC problem $\langle \Pi, I, G, C \rangle$, where the set C contains the constraints to encode the observed actions $a_1 \ldots a_k$. In PAC, these constraints are expressed through the so-called *pattern* $a_1 \ldots a_k$ constraint which can be handled by PAC-C with the addition of the extra effects of the actions in A' related to the observed actions off-the-shelf.

4.3 Metacognitive Agent

Inspired by Ganapini et al. [15], in our system, the metacognitive agent (MC) is in charge of deciding whether to accept the solution proposed by S1 or, instead, engage S2 to evaluate a better solution. Intuitively, the metacognitive assessment is twofold: On the one hand, it considers S1 confidence in its answer, but due to the fact that S1 is a machine learning approach that may have great confidence in the solution even if it is utterly wrong, the metacognitive agent also evaluates the level of correctness of System 1 in similar tasks. The pseudocode of the metacognitive process is reported in Algorithm 1. It takes in input a goal recognition instance T, the solution y_i proposed by S1, the vector p of S1 precision scores (i.e., the precision metric calculated for each fact in F on a validation set), and two thresholds τ_1, τ_2. The solution proposed by S1 is a vector of real numbers that represents a score in $[0, 1]$ for each proposition in F. First, MC calculates a score for each goal $G_i \in \mathcal{G}$. This score is obtained as the average of the GRNet output for all fluents belonging to G_i. The candidate goal $\hat{G}$ is the one with the highest score (line 1). MC also calculates the second-best candidate, $\hat{G_2}$ (line 2), that is used to define the confidence of S1 for the proposed solution. This is computed as the difference between the score of the candidate goal and the second-best candidate (line 3). We call this metric *confidence*. The intuition behind this metric is that if there is a large difference between these two goals, the neural network has identified more fluents belonging to $\hat{G}$ than belonging to $\hat{G_2}$, and therefore $\hat{G}$ is the most probable goal according to GRNet.

In order to evaluate the quality of the prediction provided by the network, we compute a metric, called *experience*. This metric indicates the network performance on problems that cover similar goals, in particular, it evaluates how often it predicted a corrected output in the past (line 12, where ϵ is a small value used to avoid division by zero). In particular for each fluent f in the candidate goal $\hat{G}$ (line 6), we calculate the average precision over only the ones with a score greater than a threshold (in our case 0.5), meaning that f is true (lines 4-12). Intuitively, this indicates how the network performs when it chooses to predict those fluents, and therefore if its prediction is reliable or not.

The confidence and experience metrics are compared with two thresholds (τ_1 divided by the number of fluents in the candidate goal and τ_2) and, if both exceed those thresholds, MC trusts the prediction made by S1, returning $\hat{G}$ as the candidate goal predicted by the whole system (lines 13-14). In our experiments, after a grid search optimization phase, we set $\tau_1 = 0.08$ and $\tau_2 = 0.8$. Otherwise, if the network has enough experience with those fluents but predicted two or more goals with similar scores, MC selects all the goals for which the scores provided by the network are in the range $[mean_{f \in \hat{G}} y_i[f] - \tau_1/|\hat{G}|, mean_{f \in \hat{G}} y_i[f]]$ (lines 15-16), these goals are then examined by S2 (line 20). Finally, if the network does not perform sufficiently on the fluents in $\hat{G}$, MC chooses to discard its predictions and all the goals in $\mathcal{G}$ are processed by S2 (lines 17-20). If S2 does not return any solution, the solution of S1 is returned instead (lines 21-25).

Domain	$\lvert A\rvert$	$\lvert F\rvert$	$\lvert G_i\rvert$	$\lvert\mathcal{G}\rvert$
BLOCKSWORLD	968	506	[4,10]	[19,21]
DEPOTS	13050	150	[2,8]	[8,10]
LOGISTICS	15154	154	[2,4]	[10,12]
ZENOTRAVEL	23724	66	[5,9]	[6,10]

Table 1: Size of A, F, $G_i \in \mathcal{G}$ and $\mathcal{G}$ in the considered GR instances for each considered domain. The interval $[x, y]$ indicates a range of integer values from x to y.

4.4 Updating System 1 using System 2

An important desideratum consists in improving the performance of S1 as the experience on the domain increases. To do that, every time the MC agent adopts S2 to solve a goal recognition instance, FSGR stores the generated optimal plans and the corresponding goals into a buffer. Plans and goals in the buffer are used to train S1. In this way, the iterative training can be seen as a fine-tuning process that enhances S1 as the system gains experience. We now describe the buffer and the training phase. The buffer is divided into two memories: M_1 and M_2. M_1 contains the L most recent samples. M_2, with a size of $2L$, maintains a selection of all the plans generated by S2 throughout the operation of FSGR. Maintaining an approximate memory of all the samples generated by S2 prevents catastrophic forgetting during the fine-tuning of S1. All the samples in M_1 are moved to M_2 after each training phase, replacing randomly selected plans/goals in M_2 whenever free space is unavailable. A new training phase starts each time M_1 fills up. During the training phase, L samples are drawn at random from the whole buffer. From each sample, a random subset of actions is selected (preserving their sequential order) to create a new observation sequence. The set of L observation sequences and the corresponding goals are used to train S1. The process is designed so that at the first training phase only the recent examples in L1 are used. As the training process repeats, the number of examples in M_2 increases. Ultimately, this will lead the system to select one-third of examples from M_1 and two-thirds from M_2. If the observations derived from the plans in M_1 contain new actions or achieve unseen fluents, the overall content of the buffer is used to train a new S1 from scratch. In our experiments, we set $L = 960$ plans (i.e., 15 training batches, each of 64 elements).

5 EXPERIMENTAL ANALYSIS

We experimentally evaluate our system and compare it with the state-of-the-art model for goal recognition PRP [29] by observing optimal plans problems. We consider four well-known benchmark domains: *i)* **BLOCKSWORLD**. The domain consists of a robotic arm that has to stack or unstack blocks, picking them one at a time, in order to obtain a desired configuration of an available set of blocks. *ii)* **DEPOTS**. The domain consists of actions to load and unload packages into trucks through hoists and move them between depots. The goals concern having the packages at certain depots. *iii)* **LOGISTICS**. In this domain, aircrafts fly between cities, trucks can move between locations within a city, and packages can be loaded into/unloaded from trucks and aircrafts. The goal is to deliver a set of packages to their delivery locations. *iv)* **ZENOTRAVEL**. This is another variant of a transportation domain where passengers have to be embarked and disembarked into aircrafts that can fly between cities at two possible speeds. The goals concern transporting (moving) all passengers (aircrafts) to their required destinations.

Of course, our system can be trained and tested using other domains. In the domains considered, we used automated planning techniques to create the (solved) GR instances for the training and test sets. Concerning the training set, for each domain, we randomly generated a large collection of (solvable) plan generation problems of different sizes. We adopted the same configuration used in the experiments of Pereira et al. [28]. In Table 1 we report the number of actions (A), fluents (F), fluents for each goal (G_i) and number of candidate goals ($\mathcal{G}$). For each problem, we computed an optimal plan for solving it, in this case, we adopted the Big Joint Optimal Landmarks Planner [11] to compute the solution. From the generated solutions, we derived the observation sequences for the training samples by randomly selecting actions from the plans preserving their relative order. The selected actions are between 30% and 70% of the plan actions. For the evaluation phase, we generated a test set made of 600 GR instances (not seen at training time) of different sizes. The procedure for obtaining problems and plans is akin to the one we described for the training set. Optimal solutions for the planning problems compiled with PAC-C are computed with the A* search guided by the h_{LM-Cut} heuristic that supports conditional effects [31], implemented in FastDownward [20]. For each optimal plan π_i, we derived three different observations by subsampling 30%, 50%, and 70% of each π_i actions. For each domain, this results in three groups of test instances, allowing us to evaluate the performance of FSGR in terms of different amounts of available observations.

The experiments were conducted on an Intel Xeon Gold 6140M 2.3 GHz processor. For calculating optimal solutions, runtime and memory were constrained to 1800 seconds and 8GB, respectively. When training the models, memory utilization was limited to 40GB. The system is evaluated in terms of two different aspects: we evaluate the GR accuracy, and we also keep track of the time taken by the systems to find a solution. For a set of test instances, the accuracy is defined as the percentage of instances whose goals are correctly identified (predicted) over the total number of instances in the test set. If, for a problem instance, the evaluated system provides k different goals with the same highest score, then, in the overall count of the solved instances, this instance has value $1/k$ if the true goal is one of these k goals, 0 otherwise.

5.1 Results

We analyzed the performance of FSGR on the test set as well as on the three subgroups of instances based on the percentage of plan observations. The performances on the three different subsets of instances of the test set in terms of GR accuracy and time are reported in Table 2. For all the considered domains, we can see that the accuracy of System 1 increases with both the number of plans used for training GRNet and the percentage of observed actions of the plan. This is consistent with the fact that the more information is used to train System 1, the better it performs. However, it is interesting to notice that the system is able to exploit the available information providing a gain either in accuracy or in time. For instance, considering 50% of the plan, the accuracy of GRNet in the DEPOTS domain changes from 24.8%, when $6k$ plans are used, to

Domain	Train plans	30% of the plan					50% of the plan					70% of the plan				
		A_{S1}	A_{S2}	A_{FSGR}	T_{S2}	T_{FSGR}	A_{S1}	A_{S2}	A_{FSGR}	T_{S2}	T_{FSGR}	A_{S1}	A_{S2}	A_{FSGR}	T_{S2}	T_{FSGR}
BLOCKSWORLD	-	-	62.5	-	9108	-	-	79.6	-	10958	-	-	88.1	-	12124	-
	0	4.7		62.5		9108	7.2		79.6		10958	7.0		88.1		12124
	6k	35.1		65.3		8364	49.5		80.6		9818	59.9		88.3		11677
	12k	42.8		66.4		5506	57.9		78.1		5811	72.3		87.9		6778
	18k	46.5		66.9		4504	62.4		77.8		4263	75.3		86.7		4320
	24k	46.3		67.7		3298	62.9		77.8		2507	77.0		87.0		2214
DEPOTS	-	-	85.4	-	4148	-	-	94.2	-	4330	-	-	97.3	-	4487	-
	0	19.2		85.4		4148	16.2		94.2		4330	16.6		97.3		4487
	6k	22.3		85.4		4148	24.8		94.2		4330	25.9		97.3		4487
	12k	45.5		81.0		2930	60.6		87.8		2580	69.3		90.2		2312
	18k	57.5		79.8		1172	74.6		89.1		378	86.3		93.0		251
	24k	58.7		82.4		837	77.1		92.3		182	87.4		95.5		73
LOGISTICS	-	-	62.4	-	10629	-	-	66.5	-	10467	-	-	67.9	-	10388	-
	0	9.7		62.4		10629	10.7		66.5		10467	9.1		67.9		10388
	6k	45.1		62.8		10375	55.8		66.6		10129	55.3		68.1		10225
	12k	52.0		69.1		5736	62.2		70.5		6164	70.5		70.8		7117
	18k	54.8		72.7		4854	65.3		75.4		4486	77.8		76.3		4658
	24k	59.1		74.5		4112	71.2		78.7		3004	81.8		79.8		3018
ZENOTRAVEL	-	-	92.7	-	4998	-	-	97.2	-	5591	-	-	98.9	-	6008	-
	0	16.0		92.7		4998	17.1		97.2		5591	18.5		98.9		6008
	6k	75.9		86.3		478	87.8		92.8		54	95.3		96.9		2
	12k	74.5		88.1		581	87.3		91.9		51	95.1		97.9		10
	18k	75.2		88.3		597	87.8		92.5		66	95.5		96.6		4
	24k	75.5		88.5		590	87.3		93.4		69	94.8		97.6		14

Table 2: Performance of FSGR in terms of accuracy (in %) and Time (in seconds) considering goal recognition problems into which the actions observed are the 30%, 50%, or 70% of the entire plan. In A_{S1}, A_{S2}, and A_{FSGR} columns, we report the accuracy of System 1 (i.e., GRNet), System 2 (i.e., PRP), and FSGR (respectively). In the T_{FSGR} column, we report the average time taken by FSGR to find a solution. For each domain, the first row reports the performance of System 2, and from the second row on, we report different stages of the incremental training of System 1.

Domain	Train plans	$\tau_1 = 0.04$						$\tau_1 = 0.08$						$\tau_1 = 0.16$					
		$\tau_2 = 0.4$		$\tau_2 = 0.8$		$\tau_2 = 0.9$		$\tau_2 = 0.4$		$\tau_2 = 0.8$		$\tau_2 = 0.9$		$\tau_2 = 0.4$		$\tau_2 = 0.8$		$\tau_2 = 0.9$	
		A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}	A_{FSGR}	T_{FSGR}
DEPOTS	0	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322
	6k	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322	92.3	4322
	12k	81.6	1884	85.2	2552	89.5	3687	83.3	1957	86.3	2607	90.0	3711	85.0	2013	87.5	2653	90.7	3721
	18k	85.0	533	84.9	556	86.6	1418	87.3	579	87.3	600	88.6	1456	88.7	631	88.7	653	89.9	1501
	24k	87.9	328	87.9	328	88.3	474	90.1	364	90.1	364	90.3	507	91.3	389	91.3	389	91.4	531

Table 3: Performance of FSGR in terms of accuracy (A_{FSGR} in %) and Time (T_{FSGR} in seconds) considering different threshold values for the confidence (τ_1) and the experience (τ_2) metrics for the DEPOTS domain.

77.1% with 24k plans. This improvement leads the metacognitive system to trust System 1 more, and thus using it more times without exploiting System 2, which is slower and more resource-demanding. The effect of this process can be seen in the T_{FSGR} column reporting the average time to compute a solution by the integrated system: the value of T_{FSGR} decreases drastically as the number of train plans increases. For instance, in DEPOTS, the time required by FSGR to find a solution changes from 4330 seconds to 182 seconds, more than twenty times less. Despite this time reduction, the system maintains a high accuracy which is always over 87%. Notice that with 70% of the observed plan, the system obtains 95.5% (just 1.8 points less than PRP) with an average time of only 73 seconds compared to the 4430 required by PRP.

An interesting result of our approach is reported in the LOGISTICS domain. For all the considered cases, it can be noticed that trusting System 1 increases the overall performance of the integrated system, which obtains an accuracy even higher than the one reached by System 2 (that we remember being the state-of-the-art). For instance, with 50% of the plan, System 2 has an accuracy of 66.5%, while the integrated system, with a fully trained GRNet, reaches 78.7%. These results are due to the fact that System 1 can provide a solution in less time than System 2, overcoming the time limit of the system. In our experiments, we set this limit to different values (i.e., 5, 10, 15, and 30 minutes). Due to the lack of space, we report the results with a time limit of 30 minutes for computing the solution of a single planning instance. Although in several cases

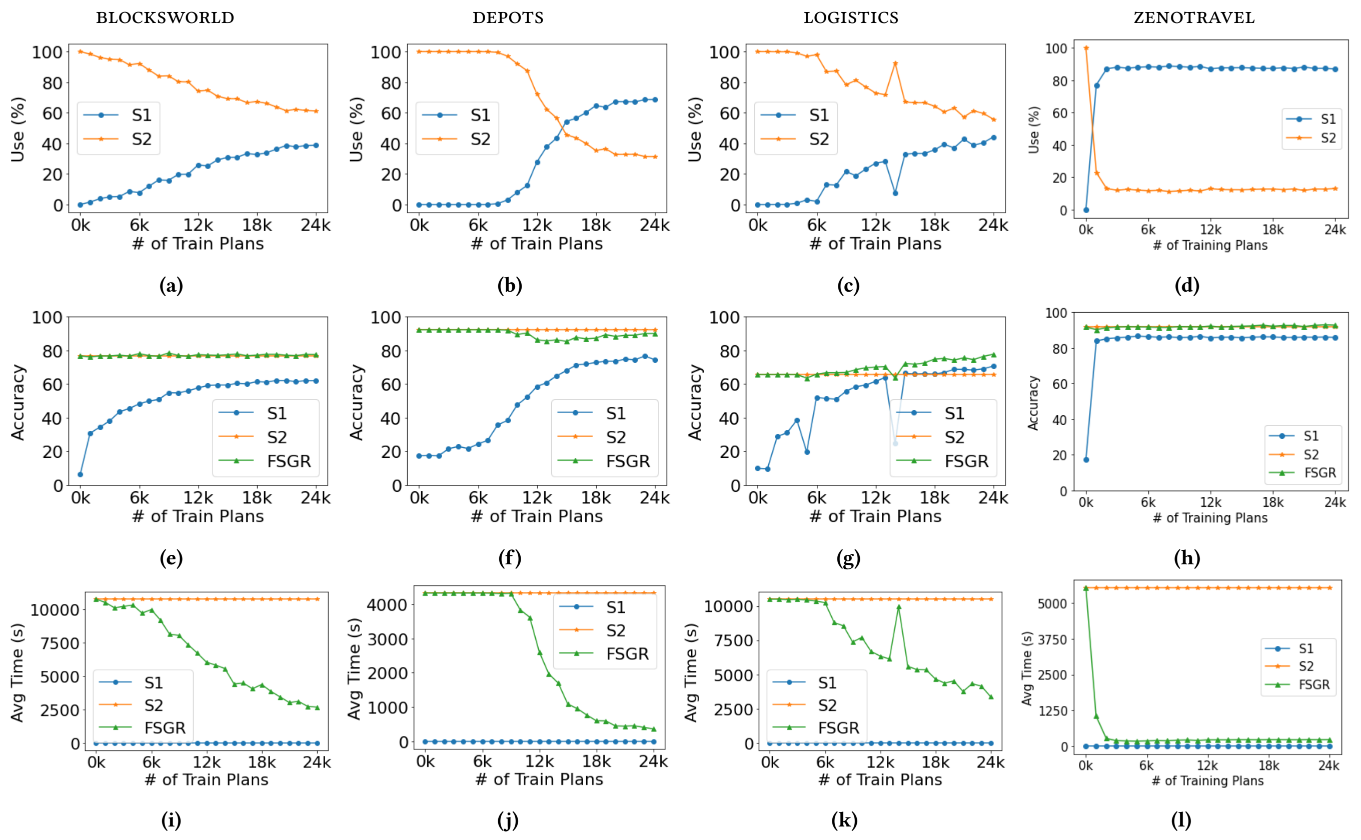

Figure 3: Average performance on BLOCKSWORLD, DEPOTS, LOGISTICS and ZENOTRAVEL domains (in columns) considering the whole test set: (first row) Use of S1 and S2; (second row) Accuracy of S1, S2, and FSGR; (third row) Time for S1, S2, and FSGR.

this time limit is not enough for S2, the system is able to compute a solution even for these cases by exploiting System 1 capabilities.

The results in Table 2 allow us to compare our system with two baselines: a system that randomly predicts a goal (in Table 2, this corresponds to the results on rows with 0 train plans, i.e. a neural network not trained which acts as a random classifier) and the state-of-the-art PRP (A_{S2} and T_{S2} columns in Table 2 respectively). We can see that when System 1 performs very poorly due to insufficient training (as in the 0k rows of Table 2 and, for instance, in the 6k row for the DEPOTS domain), the integrated system behaves almost as PRP in terms of average accuracy and time. On the contrary, with a properly trained System 1, the behavior of the integrated system shows a neglectable decrease (just a few points) in terms of accuracy with respect to PRP but with a considerable gain in terms of time. In fact, in some cases such as in BLOCKSWORLD with 24k training plans with 70%, we can even reduce the time necessary to compute a solution by more than 5 times. This shows the capability of our metacognitive system to work remarkably well in both cases, taking the best of System 1 and System 2.

In Figure 3, we graphically show the behavior of our system for the considered domains. In the Figure, each dot represents S1 performance after each fine-tuning step using 960 plans, up to 24k plans. The three rows report different plots showing three different aspects and how they vary during the incremental training of GRNet: in the first row, we report the use of Systems 1, 2 (i.e., the percentage of times the metacognitive system adopts one of the two systems). It can be noticed that, in each considered domain, S1 is more used when it is trained with a larger quantity of plans. For instance, in DEPOTS, S1 is used for 68.6% of the test instances when it is trained with 24k plans. This result is expected: in fact, a larger training set should lead to better performance for S1 and, therefore, the Metacognitive System should be more inclined to trust its predictions. In the second row we report the accuracy of S1, S2, and the overall accuracy of FSGR. In the third row, we represent the average time required to find a solution to a GR instance. It is worth noting that while Table 2 presents results categorized according to various percentages of observations, the figures presented here display the average results, considering all percentages collectively (i.e., performance refers to the whole test set). In BLOCKSWORLD, it can be noticed that the increase in the use of System 1 (see Figure 3a) corresponds to a negligible variation in accuracy (see Figure 3e), but the time taken to compute a solution decreases (see Figure 3i). A similar but more decisive behavior can be seen in ZENOTRAVEL. In fact, in this domain, even a limited number of plans allows GRNet to obtain remarkably good performance reaching 80% of use very rapidly (see Figure 3d). Again, this causes a very small variation in terms of accuracy (see Figure 3h) but reduces the time by more than 10 times (see Figure 3l). The performance in DEPOTS is similar in principle to that in ZENOTRAVEL, except for the fact that S1 needs more plans to obtain good accuracy. For LOGISTICS, Figure 3

Domain	Train	5 minutes					15 minutes				
	Plans	A_{S1}	A_{S2}	A_{FSGR}	T_{S2}	T_{FSGR}	A_{S1}	A_{S2}	A_{FSGR}	T_{S2}	T_{FSGR}
BW	-	-	69.9	-	3337	-	-	75.3	-	6864	-
	0	6.3		69.9		3337	6.3		75.3		6864
	6k	48.2		71.4		3041	48.2		76.7		6335
	12k	57.7		72.1		1868	57.7		76.4		3866
	18k	61.4		72.8		1398	61.4		76.4		2824
	24k	62.1		73.8		917	62.1		76.9		1771
DEP	-	-	90.6	-	1682	-	-	92.3	-	4265	-
	0	17.3		90.6		1682	17.3		92.3		4265
	6k	24.3		90.6		1682	24.3		92.3		4265
	12k	58.5		85.2		1023	58.5		86.3		2583
	18k	72.8		86.6		249	72.8		87.3		599
	24k	74.4		89.4		159	74.4		90.1		364
LOG	-	-	60.8	-	2135	-	-	63.4	-	5703	-
	0	9.8		60.8		2135	9.8		63.4		5703
	6k	52.1		61.3		2082	52.1		63.7		5566
	12k	61.6		66.7		1281	61.6		68.5		3437
	18k	66.0		71.5		942	66.0		73.1		2526
	24k	70.7		75.1		689	70.7		76.5		1836
ZENO	-	-	85.4	-	1398	-	-	95.0	-	3288	-
	0	17.2		85.4		1398	17.2		95.0		3288
	6k	86.3		91.5		50	86.3		92.0		109
	12k	85.6		92.2		58	85.6		92.6		131
	18k	86.2		92.0		61	86.2		92.5		136
	24k	85.9		92.7		62	85.9		93.2		137

Table 4: Performance of FSGR in terms of accuracy (in %) and Time (in seconds), considering time limits of 5 and 15 minutes for solving each problem computed by S2. In A_{S1}, A_{S2}, and A_{FSGR} columns, we report the accuracy of System 1, System 2, and FSGR (respectively). In T_{FSGR} column, we report the average time that FSGR needs to find a solution. For each domain, the first row reports the performance of System 2, from the second row instead we report different training stages of System 1.

illustrates a crucial advantage of S1 over S2, as it demonstrates that S1's quicker computation of solutions allows the entire system to produce outputs even for instances where S2 would exceed its time constraints, resulting in higher overall accuracy (see Figure 3g).

We also evaluated how sensible our system is to different threshold values of the confidence (τ_1) and the experience (τ_2) metrics. Given the limited amount of space available, in Table 3 we report the performance of the FSGR system collected only in the DEPOTS domains. Considering three different values of τ_1 and τ_2, we can see that when the neural network is not properly trained (using only 1k or 6k training problems), S1 does not have enough confidence or experience in its predictions, and therefore S2 is called most of the time (resulting in an average time of 4322 seconds to resolve a problem instance). On the other hand, when the number of training plans increases (e.g., with 12k and 18k), we can notice some differences, especially varying τ_2. For all three values of τ_1, setting a higher threshold for the experience metric (and, therefore, requesting greater precision from the neural network in some specific fluents) increases the accuracy by a few points (for instance, from 88.7 to 89.9 with $\tau_1 = 0.16$) but at the expense of the computation time (from 631 to 1501 seconds). Overall, note how the accuracy remains very high in all configurations even with lower values of both thresholds. A smaller impact can be seen when the neural network is fully trained (24k training plans). Although increasing the confidence and experience thresholds generally leads to a longer computation time without remarkable improvements in terms of accuracy, we can see that in general the S1 is generally precise enough that even setting a low τ_2 does not lead to a loss of accuracy (for instance, 87.9 with $\tau_2 = 0.4$ and $\tau_1 = 0.04$). We have similar results for LOGISTICS, while ZENOTRAVEL, which is basically more trustworthy (given the very high performance even with a small amount of training plans) even with high thresholds. In BLOCKSWORLD, higher threshold values provide a minor usage of S1 but without improvements in terms of accuracy.

Finally, Table 4 reports the performance considering different time limits (i.e., 5 and 15 minutes) to solve each problem generated by S2. For all the considered domains, we can see that the accuracy of S1 (column A_{S1}) increases as the number of training plans used increases according to what is reported in Table 2. We can also see that, as the time limit grows, both the accuracy of System 2 (columns A_{S2}) and the average time S2 needs to find a solution (columns T_{S2}) increases for almost all the domains. However, this is not the case for LOGISTICS, where the average time decreases as the time limit increases. In this domain, FSGR maintains the same properties seen in Table 2. In fact, by exploiting the new information collected through training, the metacognitive agent is able to choose S1 more often, leading to a gain in either performance or time (columns A_{FSGR} and T_{FSGR}). This is particularly noticeable when S2, due to the time limit, does not perform well; for instance, in BLOCKSWORLD with 5 minutes time limit, starting from 6k training plan, FSGR always performs better than the two separated systems both in terms of GR Accuracy and average time.

6 CONCLUSIONS AND FUTURE WORK

This paper introduces a novel approach to goal recognition which seamlessly integrates both intuitive and deliberative reasoning techniques. By proposing a dual-process model, the paper harnesses the power of fast, intuitive recognition for immediate goal identification, while employing slow, deliberate analysis for deeper understanding. This unique combination leverages machine learning techniques and planning-based reasoning, effectively modeling the dual-process system. The experimental evaluation of our system demonstrates improved accuracy and robustness w.r.t. the state-of-the-art system, especially in complex scenarios. In summary, this paper contributes to the field of goal recognition by unifying intuitive and deliberative processes, enhancing decision making, and designing intelligent systems. It bridges the gap between rapid inference and deep understanding, paving the way for advanced and proficient systems. Among the directions for future work, we intend to study effective policies to maintain the knowledge base from past experience (e.g., [17]) and BERT-based architectures [34].

ACKNOWLEDGEMENT

This work was supported by the EU H2020 project AIPlan4EU (GA 101016442), EU ICT-48 2020 project TAILOR (GA 952215), MUR PRIN project RIPER (No. 20203FFYLK), and Climate Change AI project (No. IG-2023-174).

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.

[2] Leonardo Amado, Reuth Mirsky, and Felipe Meneguzzi. 2022. Goal recognition as reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9644–9651.

[3] Leonardo Amado, Ramon Fraga Pereira, and Felipe Meneguzzi. 2023. Robust neuro-symbolic goal and plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11937–11944.

[4] Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*. 5360–5370.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research* 3 (2003), 1137–1155.

[7] Luigi Bonassi, Alfonso Emilio Gerevini, and Enrico Scala. 2022. Planning with Qualitative Action-Trajectory Constraints in PDDL. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 4606–4613.

[8] Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, and Biplav Srivastava. 2021. Thinking Fast and Slow in AI. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 15042–15046.

[9] Daniel Borrajo, Sriram Gopalakrishnan, and Vamsi K. Potluru. 2020. Goal recognition via model-based and model-free techniques. *Proceedings of the 1st Workshop on Planning for Financial Services at the Thirtieth International Conference on Automated Planning and Scheduling, FinPlan 2020* (2020).

[10] Mattia Chiari, Alfonso Emilio Gerevini, Francesco Percassi, Luca Putelli, Ivan Serina, and Matteo Olivato. 2023. Goal Recognition as a Deep Learning Task: the GRNet Approach. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 33. 560–568.

[11] Carmel Domshlak, Malte Helmert, Erez Karpas, Emil Keyder, Silvia Richter, Gabriele Röger, Jendrik Seipp, and Matthias Westphal. 2011. BJOLP: The big joint optimal landmarks planner. (2011).

[12] Yolanda E-Martín, María D. R.-Moreno, and David E. Smith. 2015. A Fast Goal Recognition Technique Based on Interaction Estimates. In *IJCAI*. AAAI Press, 761–768.

[13] M Bergamaschi Ganapini, M Campbell, F Fabiano, L Horesh, J Lenchner, A Loreggia, N Mattei, F Rossi, B Srivastava, and KB Venable. 2022. Combining fast and slow thinking for human-like and efficient decisions in constrained environments. In *Proceedings of the 16th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 2022) Co-located with (IJCLR 2022)*, Vol. 3212. 171–185.

[14] Marianna Bergamaschi Ganapini, Murray Campbell, Francesco Fabiano, Lior Horesh, Jonathan Lenchner, Andrea Loreggia, Nicholas Mattei, Francesca Rossi, Biplav Srivastava, and Kristen Brent Venable. 2021. Combining Fast and Slow Thinking for Human-like and Efficient Navigation in Constrained Environments. In *Proceedings of the Thinking Fast and Slow and Other Cognitive Theories in AI, a AAAI 2022 Fall Symposium, Westin Arlington Gateway in Arlington, Virginia, November 17-19, 2022 (CEUR Workshop Proceedings, Vol. 3332)*. CEUR-WS.org.

[15] M Bergamaschi Ganapini, Murray Campbell, Francesco Fabiano, Lior Horesh, Jon Lenchner, Andrea Loreggia, Nicholas Mattei, Francesca Rossi, Biplav Srivastava, and Kristen Brent Venable. 2022. Thinking fast and slow in AI: The role of metacognition. In *International Conference on Machine Learning, Optimization, and Data Science*. Springer, 502–509.

[16] Hector Geffner. 2018. Model-free, Model-based, and General Intelligence. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*. 10–17.

[17] Alfonso Emilio Gerevini, Alessandro Saetti, Ivan Serina, Andrea Loreggia, Luca Putelli, and Anna Roubickova. 2023. Maintenance of Plan Libraries for Case-Based Planning: Offline and Online Policies. *Journal of Artificial Intelligence Research* 78 (2023), 527–577.

[18] Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2016. *Automated Planning and Acting*. Cambridge University Press.

[19] Gautam Goel, Niangjun Chen, and Adam Wierman. 2017. Thinking fast and slow: Optimization decomposition across timescales. In *IEEE 56th Conference on Decision and Control (CDC)*. IEEE, 1291–1298.

[20] Malte Helmert. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26 (2006), 191–246.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80.

[22] Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Macmillan.

[23] Peta Masters and Sebastian Sardina. 2021. Expecting the unexpected: Goal recognition for rational and irrational agents. *Artificial Intelligence* 297 (2021), 103490. https://doi.org/10.1016/j.artint.2021.103490

[24] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. PDDL-the planning domain definition language. *Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control* (1998).

[25] Felipe Meneguzzi and Ramon Fraga Pereira. 2021. A Survey on Goal Recognition as Planning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*. 4524–4532.

[26] Reuth Mirsky, Roni Stern, Ya'akov (Kobi) Gal, and Meir Kalech. 2016. Sequential Plan Recognition. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 401–407.

[27] Ramon Pereira, Nir Oren, and Felipe Meneguzzi. 2017. Landmark-based heuristics for goal recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[28] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. 2020. Landmark-based approaches for goal recognition as planning. *Artif. Intell.* 279 (2020). https://doi.org/10.1016/j.artint.2019.103217

[29] Miquel Ramírez and Hector Geffner. 2009. Plan Recognition as Planning. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence, IJCAI 2009*. 1778–1783.

[30] Miquel Ramírez and Hector Geffner. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*. AAAI Press.

[31] Gabriele Röger, Florian Pommerening, and Malte Helmert. 2014. Optimal planning in the presence of conditional effects: Extending lm-cut with context-splitting. (2014).

[32] Francesca Rossi and Andrea Loreggia. 2019. Preferences and ethical priorities: thinking fast and slow in AI. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 3–4.

[33] Luísa R. A. Santos, Felipe Meneguzzi, Ramon Fraga Pereira, and André Grahl Pereira. 2021. An LP-Based Approach for Goal Recognition as Planning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, 11939–11946.

[34] Lorenzo Serina, Mattia Chiari, Alfonso Emilio Gerevini, Luca Putelli, and Ivan Serina. 2022. A Preliminary Study on BERT applied to Automated Planning. In *Proceedings of the 10th Italian workshop on Planning and Scheduling (IPS 2022), RCRA Incontri E Confronti (RiCeRcA 2022), and the workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (SPIRIT 2022) co-located with 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA 2022), November 28 - December 2, 2022, University of Udine, Udine, Italy (CEUR Workshop Proceedings, Vol. 3345)*. CEUR-WS.org.

[35] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. 2016. Plan Recognition as Planning Revisited. In *Proceedings of IJCAI 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press.

[36] Keith E. Stanovich and Richard F. West. 2000. Individual differences in reasoning: Implications for the rationality debate? *Behavioral and Brain Sciences* 23, 5 (Oct. 2000), 645–726.

[37] Franz A. Van-Horenbeke and Angelika Peer. 2021. Activity, Plan, and Goal Recognition: A Review. *Frontiers Robotics AI* 8 (2021).

[38] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kevin Knight, Ani Nenkova, and Owen Rambow (Eds.). The Association for Computational Linguistics, 1480–1489.